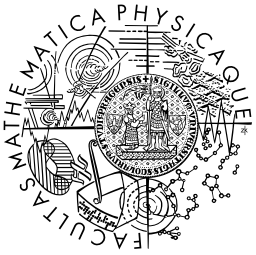


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

DIPLOMOVÁ PRÁCE

Monika Hrinčárová

Audio steganografie a IP telefonie

Katedra algebry

Vedoucí diplomové práce:	prof. RNDr. Aleš Drápal, CSc., DSc.
Studijní program:	Matematika
Studijní obor:	Matematické metody informační bezpečnosti

Praha 2017

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Touto cestou by som rada poďakovala všetkým, ktorí ma podporovali pri písaní tejto diplomovej práci. Obzvlášť rada by som poďakovala môjmu vedúcemu práce, prof. RNDr. Alešovi Drápalovi, CSc., DSc., za venovaný čas, odbornú pomoc a dôležité pripomienky a rady, ktoré mi poskytol pri jej vypracovaní. Poďakovanie patrí aj Mgr. Milanovi Boháčkovi za jeho cenné rady s aplikáciou Skype a RNDr. Andrewovi Kozlíkovi, Ph.D. za jeho odborné rady v oblasti steganografii a odporúčanú literatúru.

Názov práce: Audio steganografie a IP telefonie

Autor: Monika Hrinčárová

Katedra: Katedra algebry

Vedúci diplomovej práce: prof. RNDr. Aleš Drápal, CSc., DSc., Katedra algebry

Abstrakt: Steganografia sa zaoberá skrývaním informácií. V tejto práci budeme tajné informácie ukrývať v paketoch posielaných počas hovoru Skype. Skype patrí k jedným z najznámejších a najpoužívanejších VoIP aplikácii. Navrhneme, popíšeme a implementujeme steganografickú metódu, pomocou ktorej budeme posielať tajnú správu počas hovoru Skype. Samotné vkladanie správy do paketov a ich extrahovanie bude využívať steganografickú metódu maticového vkladania. Odolnosť tejto metódy voči strateným paketom zvýšime samoopravnými a samosynchronizačnými kódmi. Ako samoopravné kódy použijeme binárne Hammingove (7,4)-kódy a ako samosynchronizačné kódy poslúžia T-kódy.

Kľúčové slová: VoIP Skype maticové vkladanie T-kódy Hammingove kódy

Title: Audio steganography and IP telephony

Author: Monika Hrinčárová

Department: Department of Algebra

Supervisor: prof. RNDr. Aleš Drápal, CSc., DSc., Department of Algebra

Abstract: Steganography is a technique which hides secret information. In this work, we will hide a secret information in the packets which are produced during a Skype call. Skype is one of the best known and the most widely used VoIP applications. We will propose, describe and implement a steganography method by which we will send the secret message during the Skype call. For embedding the message into packets and extracting them, we will use steganographic method called matrix encoding. To avoid packet loss, we will increase the robustness of this method by error-correcting and self-synchronising codes. As error-correcting codes, we will use the binary Hamming $(7, 4)$ -codes and for the self-synchronising, we will use T-codes.

Keywords: VoIP Skype matrix encoding T-codes Hamming encoding

Obsah

Zoznam tabuliek

Úvod

Steganografia sa zaoberá ukrývaním tajných informácií. V tejto práci budeme pracovať so steganografiou vo voice over IP. Konkrétne budeme pracovať s aplikáciou Skype a utajované správy budeme vkladať do UDP paketov, ktoré sa počas hovoru posielajú. Tajná správa je posielaná vo veľkom množstve dát. Rozpoznanie tajnej správy v Skype hovore je o to ťažšie, že je hovor šifrovaný. Pri odpočúvaní hovoru vidí útočník len šifrované dáta. Cieľom práce bude nájsť vhodnú steganografickú metódu, pomocou ktorej vložíme do paketov informácie a zároveň zostane zmena pôvodného paketu od upraveného nerozpoznateľná pre nezainteresované strany.

Komunikácia po sieti prebieha pomocou tzv. kanálu. Na prenos dát v kanáli vplýva šum. Dáta sa môžu poškodiť, prípadne aj stratiť. Pokiaľ od kanálu vyžadujeme, aby preniesol všetky dáta a nepoškodil ich, musíme to zaistiť pomocou protokolov. Budeme pracovať s protokolmi, ktoré nezabezpečujú spoľahlivý prenos. Strata pár paketov nemá pre kvalitu hovoru veľký vplyv. Ak by však tieto pakety obsahovali nami posielané tajné informácie, stratili by sme časť, prípadne aj celú tajnú správu.

Navrhneme steganografický systém, ktorý bude robustný voči nespoľahlivému prenosu. Budeme pri tom využívať vlastnosti samoopravných a samosynchronizačných kódov. Od samoopravných kódov očakávame, že chyby v dátach vrátia do pôvodnej podoby. Budeme schopní v dátach chyby zistiť a následne ich opraviť. Obnovovanie dát pre nás znamená, že odosielané správy budú musieť obsahovať určitú redundanciu. Budeme pracovať s binárnymi Hammingovými kódmi.

Samoopravné kódy nie sú vhodné v prípade, že dôjde k strate dát. Budeme pracovať so samosynchronizačnými kódmi, konkrétne s T-kódmi. T-kódy patria pod Huffmanove kódy. Pomôžu nám na rýchle zotavenie zo straty dát a zároveň dáta bezstratovo komprimujú.

Štruktúra práce je tvorená teoretickou a praktickou časťou. Kapitolu 1 venujeme podobným prácam. Popíšeme existujúce steganografické metódy vhodné pre prenos zvuku. Niektoré budú pracovať s úpravou už samotného zvukového signálu a iné s úpravou paketu. Zoznámime sa s prácami, ktoré, ako naša, budú pracovať s ukrývaním správy v paketoch programu Skype.

Popíšeme kodek G.711 používaný na kódovanie a dekódovanie analógového signálu na digitálny a opačne.

Nasledovať budú kapitoly s popisom teórie potrebnej k vlastnému steganografickému návrhu. V kapitole 2 sa zoznámime so základnými teoretickými poznatkami zo samoopravných lineárnych kódov a popíšeme Hammingove kódy. Nasledovať bude steganografia a steganografická metóda maticového vkladania.

V kapitole 3 ukážeme samosynchronizačné kódy spolu s T-kódmi. Kapitola 4 bude obsahovať popis prenosu zvuku v paketoch. Popíšeme protokoly IP a UDP. Ukážeme, ako zmenou dát Skype paketu zachováme jeho integritu.

V kapitole 5 navrhneme a implementujeme vhodnú steganografickú metódu založenú na maticovom vkladaní. Ďalej popíšeme, ako by sme vkladali správu do „ideálneho“ audio signálu.

1. Súčasný stav poznania

Táto kapitola popisuje základný prehľad steganografických techník vzniknutých za účelom vkladania správ do VoIP.

Steganografia (kapitola 2) je veda zaoberajúca sa ukrývaním správ do iných objektov tak, aby sa nevzbudilo podozrenie na prebiehajúcu komunikáciu. V našom prípade chceme vložiť tajnú správu do VoIP. Prístupy môžeme rozdeliť do dvoch základných skupín:

- tajná správa posielaná v audio signále,
- tajná správa posielaná v paketoch.

Vo všeobecnosti musíme pri vkladaní tajnej správy dbať na to, aby sme nepoškodili kvalitu zvuku a nezainteresovaná strana nedokázala rozpoznať prebiehajúcu komunikáciu. Využívame pri tom napríklad maskovanie nízkych tónov vyššími tak, aby to bolo pre ľudské ucho nedetekovateľné. Zároveň nechceme o ukryté dáta prísť manipuláciou so signálom, ako je napríklad kompresia.

Ukrývanie do audio médií môžeme rozdeliť na dve ďalšie prirodzené kategórie. Buď ukrývame správy do vopred vytvorených zvukov, hudby, ... alebo prebieha komunikácia medzi dvoma stranami v reálnom čase.

V prípade, že vytvárame vlastné objekty, máme výhodu v tom, že tretia strana nemá na porovnanie originál, voči ktorému by skúmala použitie steganografie.

Popíšme najprv posielanie tajnej správy v audio signále.

1.1 Tajná správa v audio signále

Hlasové a audio signály sú analógové dáta, pričom pakety, a všeobecne dáta v sieti, sú v digitálnej podobe. To znamená, že potrebujeme kódovať a dekódovať z analógovej reprezentácie do digitálnej a späť, aby sme boli schopní preniesť analógové dáta v digitálnej podobe. To zaisťujeme pomocou kodeku, proces sa nazýva *PCM* (z anglického Pulse Code Modulation).

Jeden z najbežnejších kodekov používaných pre digitálnu hlasovú komunikáciu je G.711 [1, 4]. G.711 je ITU štandard (International Telecommunication Union). Hlasové dáta, napríklad z VoIP, kóduje do prúdu bitov. Signál, ktorý sa kóduje, je v rozmedzí 300 - 3400 Hz. G.711 vzorkuje signál na 8000 vzoriek za sekundu, vzorkovacia frekvencia je 8 kHz. Kodek reprezentuje amplitúdu vo vzorkách ako číslo v rozmedzí 256 hodnôt, a teda každému vzorku je priradené 8 bitové číslo, ktoré ho kóduje.

Používajú sa dve kompresné funkcie pre 64 Kbps PCM, A-law a μ-law. Obe kompresie sú stratové. A-law sa primárne používa v Európe a Austrálii, μ-law v Severnej Amerike a Japonsku.

- *A-law* kóduje 12 až 13 bitov hlasových dát do 8 bitov. Tie sú v decimálnej hodnote reprezentované číslami v rozmedzí $-128, \ldots, -1, 1, \ldots 128$. Záporné hodnoty reprezentuje klasicky dvojkovým doplnkom.
- μ*-law* kóduje 14 bitov hlasových dát do 8 bitov. Ich decimálna reprezentácia sú čísla $-127, \ldots, -0, 0, \ldots 127$. Záporné hodnoty sú reprezentované ako

bit pre znamienko, 0 pre kladné číslo a 1 pre záporné, a nasleduje číslo v absolútnej hodnote. Dochádza k prekrytiu v hodnote 0, ktorá je kódovaná dvoma spôsobmi.

Prenosová rýchlosť je 8 bitov na vzorku vynásobená vzorkovacou frekvenciou 8000 vzoriek za sekundu, čo je 64000 bitov za sekundu.

Keď chceme vložiť tajnú správu do audio signálu, musíme dať pozor, aby informácia nebola poškodená kompresiou. Taktiež treba dať pozor na ovplyvňovanie zvuku. V práci [1] použili intuitívny a jednoduchý prístup a vkladali informáciu do audio signálu tak, že vôbec neovplyvnili zvuk. Navrhli tak bezstratovú steganografiu pre G.711. Využili pri tom reprezentáciu záporných čísel μ-law kompresie. Nulový byte je reprezentovaný dvoma spôsobmi, kladná nula je 00000000 a záporná nula je 10000000. Ak je bit správy 0, tak použijú kladnú nulu, ak je bit správy 1, tak použijú zápornú nulu. Týmto spôsobom vkladajú informácie do zvukových dát a zvuk zostane nezmenený.

Ďalšia možná steganografická metóda je skrývanie tajnej správy v ozvene zvuku, tzv. echo hiding [11]. Tento prístup využíva to, že ľudské ucho nedokáže rozpoznať zvuky, ktoré sú prekryté inými, hlasnejšími zvukmi. K skrytiu tajnej správy vytvorili v audio súboroch ozvenu s malým oneskorením. Oneskorenie je také, aby nebolo poznateľné uchom. Pri dekódovaní správy sa detekujú medzery medzi ozvenami a rekonštruuje sa tajná správa.

Podrobnejší popis steganografických systémov môže čitateľ nájsť v publikácii *A view on latest audio steganography techniques* [5].

1.2 Tajná správa v paketoch

Prvý steganografický systém, ktorý popíšeme, je LACK (Lost Audio Packet Steganography) [9, 12, 13]. Využíva nespoľahlivý a nespojový prenos ako svoju výhodu. Autori zámerne pozdržiavajú niektoré pakety. V nich nahradia dáta tajnou správou. Spoliehajú pri tom na to, že príliš omeškané pakety sa zahodia. Príjemca, ktorý o posielaní správy vie, tento paket nezahodí, ale extrahuje správu. Aby sa táto metóda neodhalila, treba dávať pozor, aby sa neomeškávalo veľa paketov (akceptovateľné sú 3 % pre G.711 [12]). To by znamenalo zhoršenie kvality hovoru a vzbudilo by to podozrenie.

Ďalšou metódou od tých istých autorov je HICCUPS (Hidden Communication System for Corrupted Networks) [9]. Rovnako aj tentokrát spoliehajú na nespoľahlivosť prenosu. Princíp je rovnaký ako u LACK s tým rozdielom, že miesto oneskorenia paketov sa zámerne poškodia. Nebude sedieť kontrolný súčet IP paketu. Takéto pakety sa automaticky na strane príjemcu zahadzujú, pokiaľ si príjemca nie je vedomý použitej steganografie. Ak nebude veľa poškodených paketov, chyby sa budú javiť ako prirodzené a nevzbudia podozrenie. Poškodené či stratené pakety nesmú presiahnuť určitú hranicu (cca 70 %), inak Skype zmení protokol na posielanie správ z UDP (nespoľahlivej komunikácie) na TCP (spoľahlivú komunikáciu). Bližší popis Skype a protokolov nájdeme v kapitole 4.

LACK aj HICCUPS mali vplyv na kvalitu hovoru. Technika nazvaná Protocol Steganography for VoIP application [9] ukrýva dáta v nevyužitých alebo voliteľných častiach hlavičky paketu. Nepoškodzuje sa prenos paketov a hovor zostane rovnako kvalitný.

Poslednú metódu, ktorú spomenieme je SkyDe (Skype Hide) [14]. Táto metóda vkladá tajnú správu do paketov, ktoré neobsahujú žiadne hlasové dáta, do paketov s „tichom“. Dáta v pakete nahradia zašifrovanou tajnou správou. Príjemca takéto pakety odchytí a dešifruje.

2. Steganografia

Na začiatku tejto kapitoly sa oboznámime s kódovaním. Na kódovanie môžeme pozerať ako na reprezentáciu dát. Predstavíme samoopravné kódy [18, 24, 19], ktorých znalosť využijeme v steganografii [10, 7, 11, 6].

Pokiaľ chceme zabrániť tretím stranám poznať obsah správy, tak dáta zašifrujeme. Tomu sa venuje kryptografia. Steganografia, na rozdiel od kryptografie, utajuje už samotnú existenciu dát. Teda chráni nielen informácie ukryté v dátach, ale aj komunikujúce strany. Kryptografia sa spolu so steganografiou dopĺňajú a často kombinujú.

Zoznámime sa s pojmami kód a lineárny kód a ako konkrétny príklad uvedieme Hammingove kódy [8]. Nasledovať bude popis steganografie. Popíšeme vkladanie do najnižších bitov [10, 11]. To vylepšíme pomocou maticového vkladania [17, 7]. Konkrétny príklad ukážeme na maticovom vkladaní pomocou binárnych Hammingových kódov.

2.1 Úvod do lineárnych samoopravných kódov

Ako sme už v úvode práce spomenuli, kanál, po ktorom komunikujeme, nie je stopercentne spoľahlivý. Pri prenose dát dochádza k chybám a stratám. Preto, pokiaľ prenášame dáta, musíme brať do úvahy aj šum daného kanálu. Chceme zaistiť, aby druhá strana z prijatých dát dokázala zrekonštruovať odoslané dáta. A práve k tomu nám slúžia samoopravné kódy. Pokiaľ pošleme správu po kanáli, ktorý spôsobí chyby v dátach, tak od samoopravných kódov očakávame, že nám dáta vrátia do pôvodnej podoby. To znamená, že budeme schopní v dátach chyby zistiť a následne ich opraviť. Obnovovanie dát pre nás znamená, že odosielané správy budú musieť obsahovať určitú redundanciu. Na druhej strane ale chceme cez kanál posielať čo do objemu najmenej dát.

Použitím samoopravných kódov môžeme vylepšiť spoľahlivosť komunikácie na digitálnych systémoch. V poslednej dobe je táto technika veľmi rozšírená. Jej nevýhodou je, že dáta dokáže obnoviť len v prípade, ak došlo k chybám v tolerantnej miere. To, aká je tolerantná miera, ukážeme neskôr.

Na kód sa môžeme pozerať ako na neprázdnu množinu slov nad nejakou abecedou, pričom slovo je usporiadaná neprázdna množina znakov z danej abecedy. V našej práci sa budeme zaoberať len blokovými samoopravnými kódmi. Ich využitie v steganografii ukážeme pri maticovom vkladaní, podkapitola 2.4.

Definícia 1. Blokový kód nad abecedou Σ *je ľubovoľná neprázdna množina slov* $\mathcal{C} \subseteq \Sigma^n$. Dĺžka blokového kódu *je ľubovoľné číslo* $n \in \mathbb{N}^+$, $\mathbb{N}^+ = \mathbb{N} \setminus \{0\}$ *a* veľkosť blokového kódu *je počet jeho prvkov, označujeme* $|\mathcal{C}|$. *Prvky množiny* $\mathcal{C}$ *nazývame* kódové slová. *Nech* $|\Sigma| = q$, *potom hovoríme o* q-árnom *blokovom kóde.*

Spravidla sú slová tvorené m-ticou prvkov z abecedy Σ. Tie kódujeme na kódové slová tvorené n-ticou prvkov z abecedy Σ. V našom prípade bude $\Sigma = \{0,1\}$, hovoríme o *binárnom blokovom kóde.* Pokiaľ nepovieme inak, blokový kód bude pre nás vždy znamenať binárny blokový kód.

Kódovanie pomocou blokového kódu je prosté zobrazenie $f : \Sigma^m \to \Sigma^n$ pre pevne dané $n, m \in \mathbb{N}^+$. Dekódovanie je inverzná operácia ku kódovaniu.

Definícia 2. Systematický kód *je blokový kód dĺžky n nad abecedou Σ, kde prvých m znakov prenáša informáciu a zvyšných $v = n - m$ sa používa na zistenie a opravenie chyby.* Redundancia R *je pomer počtu použitých binárnych znakov k minimálnemu počtu potrebnému k prenosu rovnakej informácie,* $\mathsf{R} = \frac{n}{m}$.

Redundancia nám slúži k meraniu efektivity kódu z pohľadu prenosu informácie.

2.1.1 Hammingove kódy

Hammingove kódy (definícia 8) sú blokové *lineárne kódy* a platí, že $\Sigma = \mathbb{F}_q$, kde $\mathbb{F}_q$ je konečné pole. Slová dĺžky k kódujeme na slová dĺžky n. Na lineárne kódy sa môžme pozerať ako na vektorový podpriestor $\mathbb{F}_q^n$ dimenzie k.

Ako sme už spomenuli, v prijatých dátach budeme chyby zisťovať a opravovať. Nasledujúci príklad nám dá základnú predstavu, čo pre nás znamená oprava a zisťovanie chyby.

Príklad. Predstavme tzv. *paritný kód*. Paritný kód je kód, ktorý je schopný odhaliť jednu chybu, nie je však schopný opravy. Ako napovedá názov, tento kód slúži ku kontrole parity kódového slova. Nezabúdajme, že pracujeme s binárnymi kódmi. Predstavme si, že chceme odoslať slovo $\mathbf{m}$. Paritný kód vezme slovo $\mathbf{m} = (m_1, \ldots, m_{n-1})$ a na jeho koniec pridá *paritný bit* b. Tým vznikne kódové slovo $\mathbf{c} = (c_1, \ldots, c_n) = (m_1, \ldots, m_{n-1}, b)$.

Paritný bit je rovný nule, pokiaľ počet jednotiek v slove $\mathbf{m}$ je párny, inak priradíme paritnému bitu jednotku. Prijmime slovo $\mathbf{c}' = (c'_1, \ldots, c'_n)$ a spočítajme paritu $b' = c'_1 \oplus \cdots \oplus c'_{n-1}$. Ak $b' \neq c'_n$, tak vieme, že $\mathbf{c}'$ sme prijali s aspoň jednou chybou. Nevieme však, v akom bite chyba nastala. Ak $b' = c'_n$, tak predpokladáme, že $\mathbf{c} = \mathbf{c}'$. Pritom rovnosť nastane aj v prípade, že k chybe došlo párny počet krát. Preto hovoríme, že paritný kód vie odhaliť jednu chybu a nemá schopnosť opravy. Redundancia v kóde je $\mathsf{R} = \frac{n}{n-1} = 1 + \frac{1}{n-1}$. Čím väčšie je n, tým je redundancia nižšia. Musíme si uvedomiť, že zvyšovaním n zvyšujeme aj pravdepodobnosť, že nastane chyba.

Takýto blokový kód sa nazýva *párny paritný kód*. Analogicky existuje aj *nepárny paritný kód*. Pri nepárnom paritnom kóde priradíme paritnému bitu nulu, pokiaľ počet jednotiek v slove $\mathbf{m}$ je nepárny.

Pokiaľ nepovieme inak, pod paritným kódom budeme mať vždy na mysli párny paritný kód.

Hammingove kódy, na rozdiel od paritných kódov, sú schopné opraviť jednu chybu.

Vráťme sa k tomu, že sú dáta tvorené 0 a 1. Pri každom bite môže dôjsť k chybe s nejakou pravdepodobnosťou p. Teda namiesto odoslaného slova $\mathbf{c}$ prijmeme slovo $\mathbf{c}'$. Rozdiel medzi týmito dvoma slovami charakterizujme pomocou Hammingovej vzdialenosti.

Definícia 3. Hammingova vzdialenosť *je* $\mathsf{D}(\mathbf{c},\mathbf{c}') = |\{i : c_i \neq c'_i, 1 \leq i \leq n\}|$, *kde* $\mathbf{c} = (c_1, \ldots, c_n)$, $\mathbf{c}' = (c'_1, \ldots, c'_n) \in \Sigma^n$.

Definícia 4. Minimálna vzdialenosť *kódu* $\mathcal{C}$ *je* $\min\limits_{\substack{\mathbf{c},\mathbf{c}' \in \mathcal{C} \\ \mathbf{c} \neq \mathbf{c}'}} \mathsf{D}(\mathbf{c},\mathbf{c}')$.

Definícia 5. Váha *slova* $\mathbf{c} = (c_1, \ldots, c_n) \in \Sigma^n$ *je* $|\mathbf{c}| = |\{i : c_i \neq 0, 1 \leq i \leq n\}|$.

Hammingova vzdialenosť je metrika [19].

Definujme *guľu* o polomere r okolo slova $\mathbf{c} \in \mathcal{C}$ ako všetky body, ktorých vzdialenosť od $\mathbf{c}$ je maximálne r, nastalo maximálne r chýb: $\mathsf{S}(\mathbf{c},r) = \{\mathbf{c}' \in \Sigma^n; \mathsf{D}(\mathbf{c},\mathbf{c}') \leq r\}$.

Nasledujúce dve tvrdenia popisuje obrázok 2.1.

Tvrdenie 1. *Binárny lineárny kód $\mathcal{C}$ vie zistiť až k chýb v kódovom slove práve vtedy, keď pre minimálnu vzdialenosť d kódu $\mathcal{C}$ platí, že $d \geq k+1$.*

Dôkaz. Predpokladajme, že platí $d \geq k+1$. Chceme ukázať, že dokážeme zistiť k chýb v kódovom slove. Nech pri prenose kódového slova $\mathbf{c}^1 \in \mathcal{C}$ došlo k maximálne k chybám a prijali sme slovo $\mathbf{c}'$, $\mathsf{D}(\mathbf{c}',\mathbf{c}^1) \leq k$. Zároveň z predpokladu vieme, že pre každé dve kódové slová platí: $\forall \mathbf{c}^1, \mathbf{c}^2 : \mathsf{D}(\mathbf{c}^1, \mathbf{c}^2) \geq k+1$. Z toho vyplýva, že $\mathbf{c}' \notin \mathcal{C}$, a teda vieme zistiť k chýb.

Dokážme opačnú implikáciu. Nech v kódovom slove dokážeme odhaliť k chýb. Potom musí platiť $d \geq k+1$. Postupujme sporom a predpokladajme, že $d \leq k$. Teda $\exists \mathbf{c}^1, \mathbf{c}^2 \in \mathcal{C} : \mathsf{D}(\mathbf{c}^1, \mathbf{c}^2) = k$. Odošlime kódové slovo $\mathbf{c}^1 \in \mathcal{C}$ a nech nastane k chýb a prijmeme slovo $\mathbf{c}^2 \in \mathcal{C}$. Chybu neodhalíme, čo je spor s predpokladom.

□

Tvrdenie 2. *Binárny lineárny kód $\mathcal{C}$ vie opraviť až k chýb v kódovom slove práve vtedy, keď pre minimálnu vzdialenosť d kódu $\mathcal{C}$ platí, že $d \geq 2k+1$.*

Dôkaz. Nech $d \geq 2k+1$. Ukážeme, že potom dokážeme opraviť k chýb v kódovom slove. Odošlime slovo $\mathbf{c}^1 \in \mathcal{C}$ a prijmime slovo $\mathbf{c}'$, pričom nastalo maximálne k chýb, $\mathsf{D}(\mathbf{c}', \mathbf{c}^1) \leq k$. Ak by slovo $\mathbf{c}'$ bolo od ľubovolného iného kódového slova vzdialené maximálne o k, $\forall \mathbf{c}^2 \in \mathcal{C}, \mathbf{c}^1 \neq \mathbf{c}^2 : \mathsf{D}(\mathbf{c}', \mathbf{c}^2) \leq k$, potom z vlastnosti, že Hammingova vzdialenosť je metrika, platí: $\mathsf{D}(\mathbf{c}^1, \mathbf{c}^2) \leq \mathsf{D}(\mathbf{c}^1, \mathbf{c}') + \mathsf{D}(\mathbf{c}', \mathbf{c}^2) \leq k + k = 2k$. To je spor s predpokladom $d \geq 2k+1$. Teda platí, že $\forall \mathbf{c}^2 \in \mathcal{C}, \mathbf{c}^1 \neq \mathbf{c}^2 : \mathsf{D}(\mathbf{c}', \mathbf{c}^2) \geq k+1$. Najbližšie kódové slovo je $\mathbf{c}^1$, a teda sme schopní opraviť k chýb.

Dokážme opačnú implikáciu. Nech dokážeme opraviť k chýb. Ukážeme, že potom pre minimálnu vzdialenosť kódu platí $d \geq 2k+1$. Postupujme sporom. Nech $d \leq 2k$. Existujú kódové slová $\mathbf{c}^1, \mathbf{c}^2 \in \mathcal{C} : \mathsf{D}(\mathbf{c}^1, \mathbf{c}^2) = 2k$. Nech v kódovom slove $\mathbf{c}^1$ nastane k chýb a prijmeme slovo $\mathbf{c}'$. Potom $\mathsf{D}(\mathbf{c}^1, \mathbf{c}') = k = \mathsf{D}(\mathbf{c}', \mathbf{c}^2)$. Slovo $\mathbf{c}'$ nevieme jednoznačne odkódovať, a teda nie sme schopní opravy k chýb, čo je spor s predpokladom.

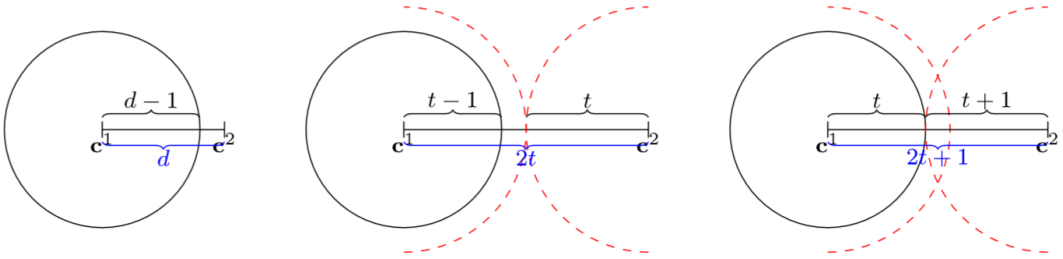
□

Lineárny kód $\mathcal{C}$ označujeme aj ako $(n,k)_q$-*kód*, kde n je dĺžka kódu, $k = \log_q |\mathcal{C}|$ a $q = |\Sigma|$. Pre binárne kódy budeme veľkosť abecedy vynechávať a hovoriť o (n,k)-kódoch.

Platí, že pre každý $(n,k)_q$-kód $\mathcal{C}$ existuje k-tica vektorov dĺžky n, ktorá ho generuje.

Definícia 6. Generujúca matica $(n,k)_q$*-kódu* $\mathcal{C}$ *je matica* G *typu* $k \times n$*, ktorej riadky generujú kód* $\mathcal{C}$*.*

Každá generujúca matica $k \times n$ sa dá pomocou Gaussovej eliminačnej metódy a zmenou poradia stĺpcov transformovať do tvaru $(\mathsf{I}_k \; \mathsf{A})$, kde I_k označuje jednotkovú maticu rádu k. Matici v tomto tvare hovoríme, že je systematická, alebo že je *v štandardnom tvare*.



Obr. 2.1: Zisťovanie a oprava chýb.

Definícia 7. Kontrolná matica $(n,k)_q$*-kódu* $\mathcal{C}$ *je matica* H *typu* $(n-k) \times n$, *pre ktorú platí:* $\mathbf{c} \in \mathcal{C} \Leftrightarrow \mathsf{H}\mathbf{c}^\top = 0$. $\mathsf{H}\mathbf{c}^\top$ *nazveme* syndróm *slova* $\mathbf{c}$.

Definícia Hammingových kódov je nasledovná:

Definícia 8. *Binárny lineárny kód s kontrolnou maticou* H, *ktorej stĺpce sú tvorené binárnou reprezentáciou čísiel* $1, 2, \ldots, 2^i - 1$, $i \in \mathbb{N} : i \geq 2$, *nazývame* Hammingov $(2^i - 1, 2^i - 1 - i)$-kód.

Ako sme už spomenuli, kódovanie je len transformácia slova na kódové slovo. Majme generujúcu maticu G kódu $\mathcal{C}$. Slovo $\mathbf{m}$ kódujeme na kódové slovo $\mathbf{c}$ pomocou generujúcej matice G nasledovne: $\mathbf{m}.\mathsf{G} = \mathbf{c}$.

Pusťme sa do dekódovania Hammingových kódov. Majme odoslané slovo $\mathbf{c}$ a prijaté slovo $\mathbf{c}'$. Označme *chybové slovo* ako rozdiel medzi prijatým a odoslaným slovom, $\mathbf{e} = \mathbf{c}' - \mathbf{c}$. Pre syndróm prijatého slova platí:

$$\mathsf{H}\mathbf{c}'^\top = \mathsf{H}\mathbf{e}^\top + \mathsf{H}\mathbf{c}^\top = \mathsf{H}\mathbf{e}^\top.$$

Syndróm prijatého a chybového slova je rovnaký. Pokiaľ $\mathsf{H}\mathbf{c}'^\top = 0$, tak $\mathbf{c}' \in \mathcal{C}$ a predpokladáme, že $\mathbf{c} = \mathbf{c}'$. Nech $\mathsf{H}\mathbf{c}'^\top \neq 0$. Chceme nájsť $\mathbf{c} \in \mathcal{C}$ najbližšie k $\mathbf{c}'$, $|\mathbf{c}' - \mathbf{c}| = |\mathbf{e}|$ čo najmenšie. To je to isté, ako hľadanie chybového slova $\mathbf{e}$, pre ktoré platí $|\mathbf{e}|$ je najmenšie a zároveň $\mathbf{c}' - \mathbf{e} = \mathbf{c} \in \mathcal{C}$. Druhá podmienka platí práve vtedy, keď $\mathsf{H}\mathbf{c}'^\top = \mathsf{H}\mathbf{e}^\top$. Pre $(n,k)_q$-kód existuje q^{n-k} rôznych nenulových syndrómov. Algoritmus je nasledovný:

1. vypočítame syndróm $\mathbf{s} = \mathsf{H}\mathbf{c}'^\top$,
2. nájdeme slovo $\mathbf{e}$, $\mathsf{H}\mathbf{e}^\top = \mathbf{s}$ a zároveň $|\mathbf{e}|$ je minimálna,
3. získame $\mathbf{c} = \mathbf{c}' - \mathbf{e}$,
4. nájdeme $\mathbf{m}$ také, že $\mathbf{c} = \mathbf{m}\mathsf{G}$.

Pri Hammingových kódoch je $\mathbf{e}$ jednoznačne dané (najbližšie kódové slovo je jednoznačne určené), označujeme ho ako *reprezentant syndrómu* $\mathbf{s}$. Ak došlo k chybe na i-tom mieste, $1 \leq i \leq n$, tak $\mathbf{s}$ je i-ty stĺpec matice H. Ak je matica G v štandardnom tvare, tak $\mathbf{m}$ je prvých m symbolov $\mathbf{c}$.

Skonštruujme Hammingov (7,4)-kód. K tomu postačí skonštruovať jeho generujúcu maticu. Z definície 8 vyplýva, že kontrolná matica tohto kódu je:

$$\mathsf{H} = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Z vlastnosti $\mathsf{H}\mathsf{G}^\top = 0$ a použitím Gaussovej eliminačnej metódy dostávame generujúcu maticu G Hammingovho (7,4)-kódu v štandardnom tvare:

$$\mathsf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

2.2 Úvod do steganografie

Steganografia je technika na ukrývanie dát a utajovania priebehu tajnej komunikácie. Steganografia bola veľmi obľúbená už v minulosti. Napríklad bola otrokom oholená hlava a vytetovala sa na ňu správa. Po dorastení vlasov bola správa ukrytá a otrok mohol nepozorovane preniesť tajnú správu. Ďalšími príkladmi sú neviditeľný atrament, ukrytá informácia v prvých písmenách každého riadku a ďalšie. V tejto práci sa budeme zaoberať ukrývaním existencie dát v digitálnom svete.

Princíp ukrytia správ zostáva rovnaký. Tajnú správu skrývame medzi iné, už existujúce dáta. Bežne používame rôzne médiá ako napríklad obrázky, videá, texty, programy alebo v našom prípade audio pakety. Práve multimediálne súbory sú vhodné vďaka ich veľkosti a redundancii.

Na úvod predstavme pojmy, s ktorými budeme v tejto kapitole pracovať. Objekt, do ktorého vkladáme utajované správy, budeme nazývať *nosič*. Množinu všetkých nosičov označme $\mathcal{C}$ a množinu všetkých správ $\mathcal{M}$. Množinu všetkých kľúčov označme $\mathcal{K}$. Keď do nosiča ukryjeme správu, stane sa z neho *stego-objekt*. Vkladanie správy do nosiča a extrahovanie správy zo stego-objektu prebieha pomocou *steganografického nástroja*.

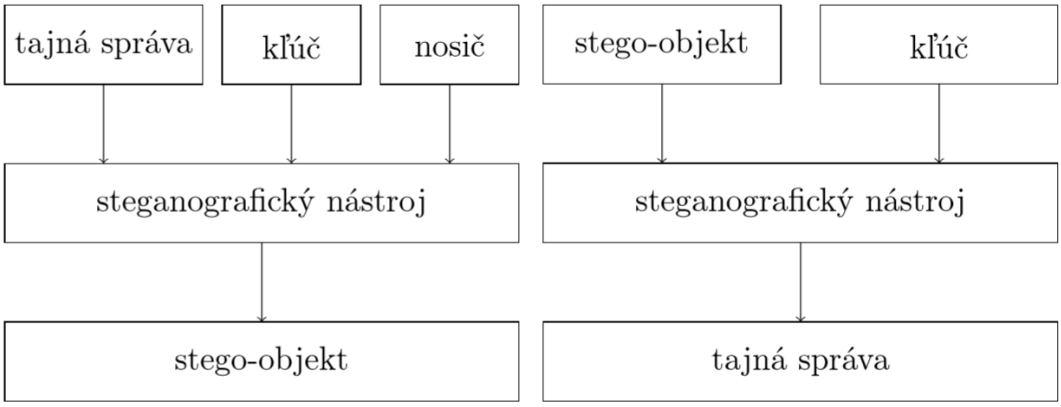
Steganografický nástroj je tvorený dvoma algoritmami. *Algoritmom na vkladanie* a *algoritmom na extrakciu* správy. Prvý algoritmus slúži na vloženie tajnej správy do nosiča. Vstupom do algoritmu je nosič, správa a kľúč (nazývaný aj *stego-kľúč*). Výstupom je stego-objekt: $\mathrm{Emb}(\mathbf{x}, \mathbf{m}, \mathbf{k}) = \mathbf{y}$, kde $\mathbf{x} \in \mathcal{C}$ je nosič, $\mathbf{m} \in \mathcal{M}$ je tajná správa, $\mathbf{k} \in \mathcal{K}$ je zdieľaný stego-kľúč a $\mathbf{y} \in \mathcal{C}$ je stego-objekt, ktorý vznikne vložením správy do nosiča. Pri vkladaní správy musíme brať do úvahy typ nosiča a pozmeňovať ho tak, aby bol pre nezainteresované strany nerozoznateľný od stego-objektu.

Algoritmus na extrakciu správy dostane na vstupe stego-objekt a zdieľaný stego-kľúč. Výstupom je vložená správa: $\mathrm{Ext}(\mathbf{y}, \mathbf{k}) = \mathbf{m}$, kde $\mathbf{m} \in \mathcal{M}$ je správa, $\mathbf{y} \in \mathcal{C}$ je stego-objekt, ktorý vznikol vložením správy $\mathbf{m}$ do nosiča a $\mathbf{k} \in \mathcal{K}$ je zdieľaný stego-kľúč. Platí, že:

$$\forall \mathbf{x} \in \mathcal{C}, \forall \mathbf{m} \in \mathcal{M}, \forall \mathbf{k} \in \mathcal{K} : \mathrm{Ext}(\mathrm{Emb}(\mathbf{x}, \mathbf{m}, \mathbf{k}), \mathbf{k}) = \mathbf{m}.$$

Tajná správa spolu s nosičom, stego-kľúčom, stego-objektom a steganografickým nástrojom tvoria *steganografický systém*, obrázok 2.2. Prípadne sa do steganografického systému zahrňuje aj fyzický kanál, po ktorom prebieha komunikácia.

Tak, ako aj v kryptografii, aj v steganografii platí *Kerckhoffov princíp*. Bezpečnosť steganografického systému leží na bezpečnosti zvoleného stego-kľúča, pričom sa predpokladá, že útočník má znalosť o použitom steganografickom systéme.



Obr. 2.2: Steganografický systém.

Útočníka v steganografii rozdeľujeme do troch základných kategórii:

- *pasívny útočník*, ktorý len pozoruje komunikáciu a snaží sa odhaliť, či komunikujúce strany používajú steganografiu. Tento typ útočníka nijako nezasahuje do samotnej komunikácie.
- *aktívny útočník*, na rozdiel od pasívneho útočníka, zasahuje do samotnej komunikácie. Modifikuje posielané správy (text, obrázky, audio, pakety, ...) tak, aby znemožnil použitie steganografie. Typickými príkladmi sú napríklad parafrázovanie textu, zamieňanie slov za ich synonymá, menenie veľkosti obrázku, prípadne jeho orezanie, ...
- *škodný útočník* tiež aktívne zasahuje do prebiehajúcej komunikácie. Snaží sa uhádnuť použitý steganografický systém a tým oklamať komunikujúce strany tak, aby sa samé odhalili.

Steganografický systém je prelomený, ak dôjde k odhaleniu prebiehajúcej tajnej komunikácie. Nie je k tomu nutné odhaliť samotnú tajnú správu.

Predstavme najzákladnejšiu metódu steganografie, vkladanie do najnižšieho bitu.

2.3 Vkladanie do najnižšieho bitu

Pokiaľ chceme ukryť informáciu do bitovej reprezentácie objektu, tak najjednoduchšou metódou je vkladanie do najnižšieho, teda najmenej významného bitu. Meníme najnižší bit z každého bytu tak, aby sme z jednotlivých najnižších bitov vedeli zrekonštruovať tajnú správu. Vkladanie do najnižšieho bitu budeme nazývať aj LSB vkladanie (z angl. Least Significant Bit).

Sú dve techniky LSB vkladania, tzv. LSB replacement a LSB matching. Voľne by sme ich mohli preložiť ako LSB nahradzovanie a LSB korešpondovanie, ale ich anglický ekvivalent viac vystihuje podstatu.

LSB replacement mení najnižší bit z každého bytu nosiča tak, aby tvorili tajnú správu.

Príklad. Nech chceme poslať správu $\mathbf{m} = (m_1, m_2, \ldots, m_n) \in \{0,1\}^n$ a nech $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ je prvých n bytov nosiča, $n \in \mathbb{N}^+$. Správu vložíme do nosiča nahradením najnižšieho bitu za bit správy:

- $\mathbf{y}_i := \mathbf{x}_i$, pre $\forall i : 1 \leq i \leq n$,
- $\text{LSB}(\mathbf{y}_i) := m_i$, $1 \leq i \leq n$,

kde $\mathbf{y}_i$ je i-ty byte stego-objektu. Extrahovanie správy je jednoduché, $m_i := \text{LSB}(\mathbf{y}_i)$, $1 \leq i \leq n$.

V najlepšom prípade nepotrebujeme zmeniť žiaden bit, v najhoršom všetky. Priemerný počet zmien je:

$$0.5n = \frac{n}{2},$$

kde $n \in \mathbb{N}^+$ je dĺžka správy a 0.5 je pravdepodobnosť, že potrebujeme zmeniť bit.

LSB matching pracuje podobne ako LSB replacement. Namiesto zmeny najnižšieho bitu pripočítavame, prípadne odčítavame k danému bytu jednotku. Preto sa LSB matching nazýva aj ± 1 vkladanie.

Príklad. Majme správu $\mathbf{m} = (m_1, m_2, \ldots, m_n) \in \{0,1\}^n$ a nech $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ je prvých n bytov nosiča, $n \in \mathbb{N}^+$. Navyše majme daný rozsah hodnôt v nosiči a hodnotu $h \in \{-1, +1\}$. Vkladanie správy prebieha nasledovne:

- ak $\text{LSB}(\mathbf{x}_i) = m_i$, potom $\mathbf{y}_i := \mathbf{x}_i$,
- ak $\text{LSB}(\mathbf{x}_i) \neq m_i$, potom:
  - náhodne zvolíme $h \in \{-1, +1\}$ tak, aby $\mathbf{x}_i + h$ nepresiahlo predom daný rozsah hodnôt v nosiči [1],
  - $\mathbf{y}_i := \mathbf{x}_i + h$,

kde $\mathbf{y}_i$ je i-ty byte stego-objektu a $1 \leq i \leq n$. Extrahovanie správy je totožné ako pri LSB replacement.

Ak je $\mathbf{x}_i = (0, \ldots, 0)$, tak h musí byť nutne $+1$. Naopak, ak $\mathbf{x}_i = (1, \ldots, 1)$, tak h musí byť nutne -1. Rozsah hodnôt je daný charakterom dát v nosiči. Môže sa jednať o reprezentovanie pixelu v obrázku, amplitúdu v audiu a podobne.

LSB vkladania je citlivé na transformáciu stego-objektu (rotácia, kompresia, filter, ...). Pri transformácii môže dôjsť až k úplnej strate správy.

Uvedomme si, že ak je najnižší bit rôzny od bitu správy, tak LSB replacement vyžaduje jednu zmenu na jeden bit správy, ale LSB matching vyžaduje aspoň jednu zmenu.

Predstavme maticové vkladanie. Jeho veľkou výhodou oproti LSB vkladaniu je to, že zmenou jedného prvku nosiča, nemusí byť nutne binárny, môžme prenášať niekoľko bitov informácie.

[1] Typickým príkladom je obrázok, ktorého pixely sú reprezentované pomocou RGB hodnôt. Rozsah hodnôt v nosiči je v takom prípade daný hodnotami $0 - 255$.

2.4 Maticové vkladanie

Maticové vkladanie, na rozdiel od LSB vkladania, dokáže pri menšom množstve zmien v nosiči preniesť rovnako veľa informácie. Je to metóda, ktorá zvyšuje bezpečnosť steganografickej schémy tak, že minimalizuje počet zmien v nosiči potrebných na ukrytie tajnej správy.

Zaveďme značenie, ktoré bude platiť po zvyšok tejto kapitoly. Budeme pracovať nad konečným poľom $\Sigma = \mathbb{F}_q$. Typicky je $\Sigma = \{0, 1\}$. Nosič je $(p_1, p_2, \ldots, p_n) \in \mathcal{G}^n$, kde $\mathcal{G}$ je množina všetkých možných hodnôt jednotlivých prvkov nosiča. Predpokladajme, že máme funkciu $\mathrm{s} : \mathcal{G} \to \mathbb{F}_q$, ktorá priraďuje prvkom nosiča z $\mathcal{G}$ prvky konečného poľa $\mathbb{F}_q$. Najčastejšie je s funkcia LSB alebo zvyšok po delení q.

Označme $\mathbf{x} = (x_1, x_2, \ldots, x_n) = (\mathrm{s}(p_1), \mathrm{s}(p_2), \ldots, \mathrm{s}(p_n)) \in \mathbb{F}_q^n$, $n \in \mathbb{N}^+$. Keď budeme hovoriť o nosiči $\mathbf{x}$, budeme mať vždy namysli nosič po transformácii funkciou s.

Správu budeme značiť $\mathbf{m} = (m_1, m_2, \ldots, m_l) \in \mathbb{F}_q^l$, $l \in \mathbb{N}^+$. Stego-objekt, ktorý vznikne vložením správy $\mathbf{m}$ do nosiča $\mathbf{x}$ je $\mathbf{y} = (y_1, y_2, \ldots, y_n) \in \mathbb{F}_q^n$, $n \in \mathbb{N}^+$. Ďalej H bude matica typu $l \times n$, pre ktorú platí, že $\mathsf{H}\mathbf{y}^\top = \mathbf{m}^\top$. Na nosič $\mathbf{x}$ a stego-objekt $\mathbf{y}$ môžeme pozerať ako na slová dĺžky n a na správu $\mathbf{m}$ ako na slovo dĺžky l.

Extrakciu správy zo stego-objektu sme už spomenuli. Majme danú maticu H, potom správu $\mathbf{m}$ získame zo stego-objektu $\mathbf{y}$ ako $\mathbf{m}^\top = \mathsf{H}\mathbf{y}^\top$. Ukážeme, ako musí prebiehať vkladanie správy, aby platilo, že $\mathsf{H}\mathbf{y}^\top = \mathbf{m}^\top$. Ak platí $\mathsf{H}\mathbf{x}^\top = \mathbf{m}^\top$, tak položíme $\mathbf{y} = \mathbf{x}$. Ak $\mathsf{H}\mathbf{x}^\top \neq \mathbf{m}^\top$, tak hľadáme také slovo $\mathbf{e}$, aby platilo $\mathsf{H}(\mathbf{x}+\mathbf{e})^\top = \mathbf{m}^\top$ a zároveň, aby bolo $\mathbf{e}$ čo najmenšej váhy, spravili sme čo najnižší počet zmien v nosiči. Stego-objekt $\mathbf{y} = \mathbf{x} + \mathbf{e}$.

Nech $\mathcal{C}$ je blokový lineárny $(n, k)_q$-kód. H je kontrolná matica kódu $\mathcal{C}$. Platí, že dĺžka správy $\mathbf{m}$ je $l = n - k$ a n je dĺžka kódu $\mathcal{C}$ ako aj dĺžka nosiča. Zaveďme pojem pokrývajúci polomer kódu.

Definícia 9. *Nech $\mathcal{C}$ je blokový lineárny $(n, k)_q$-kód.* Pokrývajúci polomer R kódu $\mathcal{C}$ *je* $\mathsf{R} = \max\limits_{\mathbf{z} \in \mathbb{F}_q^n} \mathsf{D}(\mathbf{z}, \mathcal{C})$, *kde* $\mathsf{D}(\mathbf{z}, \mathcal{C}) = \min\limits_{\mathbf{c} \in \mathcal{C}} \mathsf{D}(\mathbf{z}, \mathbf{c})$ *je Hammingova vzdialenosť slova* $\mathbf{z}$ *od kódu* $\mathcal{C}$.

Na dôkaz nasledujúcej vety využijeme: $\forall \mathbf{x} \in \mathbb{F}_2^n$, pre ktoré $\mathsf{H}\mathbf{x}^\top = \mathsf{H}\mathbf{e}^\top$, kde $\mathbf{e}$ je reprezentant daného syndrómu s najnižšou váhou:

$$\mathsf{R} = \max_{\mathbf{z} \in \mathbb{F}_q^n} \mathsf{D}(\mathbf{z}, \mathcal{C}) \geq \mathsf{D}(\mathbf{x},\mathcal{C}) = \min_{\mathbf{c} \in \mathcal{C}} |\mathbf{x} - \mathbf{c}| = |\mathbf{e}|. \tag{2.1}$$

R je pokrývajúci polomer kódu $\mathcal{C}$. Slovo $\mathbf{e}$ je od najbližšieho kódového slova vzdialené o $|\mathbf{e}|$, hodnota pokrývajúceho polomeru R je aspoň $|\mathbf{e}|$.

Veta o maticovom vkladaní hovorí, že vieme použiť blokový lineárny $(n, k)_q$-kód k preneseniu $l = n - k$ prvkov informácie z $\mathbb{F}_q$ v n prvkoch z $\mathbb{F}_q$. Pritom máme zhora ohraničený počet zmien, ktorý pri prenose nastane.

Veta 3 (O maticovom vkladaní)**.** *Nech $\mathcal{C}$ je blokový lineárny $(n, k)_q$-kód. Nech H je kontrolná matica typu $l \times n$, $l = n - k$ a R je pokrývajúci polomer kódu $\mathcal{C}$. Ďalej nech $\mathbf{m} \in \mathbb{F}_q^l$, $\mathbf{x} \in \mathbb{F}_q^n$ a nech syndróm slova $\mathbf{e}$ je $\mathbf{m} - \mathsf{H}\mathbf{x}^\top$, $\mathbf{e}$ je reprezentant syndrómu s najnižšou váhou. Potom steganografický nástroj* [2]*:*

[2] Všimnime si, že sme vynechali stego-kľúč zo vstupov do algoritmov na vkladanie a extrakciu. Platnosť vety nie je závislá na použitom stego-kľúči.

- *s algoritmom na vkladanie:* $\mathrm{Emb}(\mathbf{x}, \mathbf{m}) = \mathbf{x} + \mathbf{e} = \mathbf{y}$ *a*
- *s algoritmom na extrakciu:* $\mathrm{Ext}(\mathbf{y}) = \mathsf{H}\mathbf{y}^\top$

dokáže vložiť správu $\mathbf{m}$ *do nosiča* $\mathbf{x}$ *a pri vkladaní dôjde k maximálne* R *zmenám.*

Dôkaz. Ukážeme, že dokážeme preniesť informáciu veľkosti $n - k$ v n prvkoch z $\mathbb{F}_q$. Využijeme, že syndróm slova $\mathbf{e}$, $\mathsf{H}\mathbf{e}^\top$, je rovný $\mathbf{m} - \mathsf{H}\mathbf{x}^\top$:

$$\mathrm{Ext}(\mathrm{Emb}(\mathbf{x},\mathbf{m})) = \mathrm{Ext}(\mathbf{y}) = \mathsf{H}\mathbf{y}^\top = \mathsf{H}\mathbf{x}^\top + \mathsf{H}\mathbf{e}^\top = \mathsf{H}\mathbf{x}^\top + \mathbf{m} - \mathsf{H}\mathbf{x}^\top = \mathbf{m}.$$

Pri prenose dôjde k maximálne R zmenám, pretože:

$$\mathsf{D}(\mathbf{x}, \mathrm{Emb}(\mathbf{x}, \mathbf{m})) = \mathsf{D}(\mathbf{x}, \mathbf{y}) = \mathsf{D}(\mathbf{x}, \mathbf{x} + \mathbf{e}) = \mathsf{D}(0, \mathbf{e}) = |\mathbf{e}| \leq \mathsf{R},$$

kde funkcia D označuje Hammingovu vzdialenosť. Využili sme pri tom nerovnosť 2.1.

Máme $\mathsf{D}(\mathbf{x}, \mathrm{Emb}(\mathbf{x}, \mathbf{m})) \leq \mathsf{R}$ a teda pri prenose nastane maximálne R zmien. □

R niekedy nazývame aj *hranica skreslenia*.

Nech $\mathcal{C}$ je Hammingov kód a $\mathbf{c} \in \mathcal{C}$. Slovo $\mathbf{c}+\mathbf{e}$, kde $|\mathbf{e}| = 1$, je od najbližšieho kódového slova vzdialené 1, najbližšie kódové slovo je $\mathbf{c}$. Ak $|\mathbf{e}| = 2$, tak existuje také kódové slovo, od ktorého je slovo $\mathbf{c}+\mathbf{e}$ vzdialené 1. Vychádzame pri tom z poznatku, že minimálna vzdialenosť Hammingových kódov je 3 (podkapitola 2.1). Teda pokrývajúci polomer pre Hammingove kódy je $\mathsf{R} = 1$.

Zamerajme sa na maticové vkladanie pomocou binárnych Hammingových kódov dĺžky n a dimenzie k. Pre extrakciu tajnej správy zo stego-objektu $\mathbf{y}$ platí $\mathrm{Ext}(\mathbf{y}, \mathbf{k}) = \mathsf{H}\mathbf{y}^\top$, kde H je kontrolná matica tohto kódu a $\mathbf{k}$ je použitý stego-kľúč.

Ukážme, ako prebieha vkladanie. Ak pre nosič $\mathbf{x}$ a správu $\mathbf{m}$ platí, že $\mathsf{H}\mathbf{x}^\top = \mathbf{m}^\top$, tak $\mathbf{y} := \mathbf{x}$. V opačnom prípade hľadáme slovo $\mathbf{e}$ najnižšej váhy so syndrómom rovným $\mathbf{m} - \mathsf{H}\mathbf{x}^\top$. Uvedomme si podobnosť s dekódovaním Hammingových kódov. Nájdime taký stĺpec matice H, ktorý je rovný $\mathbf{m} - \mathsf{H}\mathbf{x}^\top$. Nech je to j-ty stĺpec matice, $1 \leq j \leq n$ a označme ho $\mathsf{H}_j = \mathbf{m} - \mathsf{H}\mathbf{x}^\top$. Taký stĺpec určite existuje, pretože stĺpce matice H obsahujú všetky čísla v binárnom zápise od 1 po n. Inak povedané, ak taký stĺpec neexistuje, nutne musel nastať prvý prípad ($\mathsf{H}\mathbf{x}^\top = \mathbf{m}^\top$). Nech $\mathbf{e}_j \in \{0,1\}^n$ označuje slovo váhy 1 s jednotkou na j-tom mieste, $1 \leq j \leq n$. Stego-objekt $\mathbf{y}$ vznikne ako nosič $\mathbf{x}$ upravený na danom j-tom mieste: $\mathbf{y} := \mathbf{x} + \mathbf{e}_j$.

Overme správnosť takejto konštrukcie vkladania:

$$\mathrm{Ext}(\mathrm{Emb}(\mathbf{x}, \mathbf{m}, \mathbf{k}), \mathbf{k}) = \mathsf{H}\mathbf{y}^\top = \mathsf{H}(\mathbf{x} + \mathbf{e}_j)^\top = \mathsf{H}\mathbf{x}^\top + \mathsf{H}\mathbf{e}_j^\top = \mathsf{H}\mathbf{x}^\top + \mathsf{H}_j = \mathbf{m}.$$

Príklad. Majme Hammingov (7,4)-kód s kontrolnou maticou:

$$\mathsf{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Nosič je $\mathbf{x} = (x_1, x_2, \ldots, x_7) \in \{0,1\}^7$ a správa je $\mathbf{m} = (m_1, m_2, m_3) \in \{0,1\}^3$. Označme $\mathbf{z} = (z_1, z_2, z_3) \in \{0,1\}^3$ slovo, pre ktoré $\mathsf{H}\mathbf{x}^\top = \mathbf{z}^\top$. Teda platí, že $z_1 = x_1 \oplus x_2 \oplus x_4 \oplus x_5$, $z_2 = x_1 \oplus x_3 \oplus x_4 \oplus x_6$ a $z_3 = x_2 \oplus x_3 \oplus x_4 \oplus x_7$.

Vytvorme tabuľku, ktorej prvé tri stĺpce sú hodnoty z_1, z_2 a z_3. Posledný stĺpec bude vyjadrovať, kde má nastať v nosiči zmena, aby platilo $\mathsf{H}\mathbf{x}^\top = \mathbf{m}^\top$.

Ak $\mathbf{z} = \mathbf{m}$, tak $\mathbf{x}$ nemením. Ak je $\mathbf{z}$ zhodné s $\mathbf{m}$ až na prvý bit, tak $\mathbf{m}$ vložíme do $\mathbf{x}$ zmenou bitu x_5. Pokiaľ je $\mathbf{z}$ s $\mathbf{m}$ odlišné vo všetkých bitoch, stačí zmeniť bit x_4, pretože ten ovplyvňuje všetky tri bity syndrómu slova $\mathbf{x}$. Rovnakou úvahou pokračujme pre všetky kombinácie správy $\mathbf{m}$. Vznikne nám nasledujúca tabuľka 2.1. Vidíme, že skutočne vystačíme s maximálne jednou zmenou v nosiči

$x_1 \oplus x_2 \oplus x_4 \oplus x_5$	$x_1 \oplus x_3 \oplus x_4 \oplus x_6$	$x_2 \oplus x_3 \oplus x_4 \oplus x_7$	invertovaný bit
m_1	m_2	m_3	
$m_1 \oplus 1$	m_2	m_3	x_5
m_1	$m_2 \oplus 1$	m_3	x_6
m_1	m_2	$m_3 \oplus 1$	x_7
m_1	$m_2 \oplus 1$	$m_3 \oplus 1$	x_3
$m_1 \oplus 1$	m_2	$m_3 \oplus 1$	x_2
$m_1 \oplus 1$	$m_2 \oplus 1$	m_3	x_1
$m_1 \oplus 1$	$m_2 \oplus 1$	$m_3 \oplus 1$	x_4

Tabuľka 2.1: Maticové vkladanie (7,4)-kódom

k zakódovaniu správy $\mathbf{m}$ do $\mathbf{x}$.

Z tabuľky 2.1 vidíme, že pokiaľ $\mathsf{H}\mathbf{x}^\top = \mathbf{m}^\top$, prenášame tri bity informácie a nepotrebujeme spraviť žiadnu zmenu v nosiči. V ostatných prípadoch vkladáme tri bity informácie pri jednej zmene.

Základným parametrom, pomocou ktorého môžeme ohodnotiť metódu steganografie, je *efektivita vkladania*, $\mathsf{e} = \frac{\mathsf{C}_e}{\mathsf{D}_a}$. C_e je *kapacita vkladania* definovaná ako $\mathsf{C}_e = \log_2 |\mathcal{M}|$. Kapacita vkladania je závislá na nosiči a vyjadruje počet bitov správy, ktoré môžem zakódovať, aby bol stego-objekt považovaný za bezpečný pri konkrétnej steganografickej schéme [6]. D_a je očakávaná hodnota distorzie (deformácie), *očakávaný počet zmien* v nosiči pri vkladaní cez všetky správy, stego-kľúče a nosiče: $\mathsf{D}_a = \mathsf{E}[\mathsf{D}(\mathbf{x}, \mathrm{Emb}(\mathbf{x}, \mathbf{m}, \mathbf{k}))]$.

Spočítajme efektivitu vkladania pre Hammingove $(n,k) = (2^l - 1, 2^l - 1 - l)$-kódy. Chcem zakódovať $l = n-k$ bitov správy do $n = 2^l - 1$ bitov nosiča. S pravdepodobnosťou $\frac{1}{2^l}$ nezmeníme žiaden z n bitov nosiča a s pravdepodobnosťou $1 - \frac{1}{2^l}$ zmeníme práve jeden bit. Očakávaný počet zmien je $0.\frac{1}{2^l} + 1.(1 - \frac{1}{2^l}) = 1 - \frac{1}{2^l}$. Kapacita vkladania je $\log_2 |\mathbb{F}_2^l| = l$. Efektivita vkladania je $\mathsf{e} = \frac{l}{1-\frac{1}{2^l}} = \frac{l2^l}{2^l-1}$. Efektivita vkladania Hammingovho (7,4)-kódu je $\mathsf{e} = \dfrac{24}{7}$.

Popísali sme základné definície a vlastnosti lineárnych samoopravných kódov. Ukázali sme, ako prebieha kódovanie a dekódovanie Hammingových kódov. Predstavili sme steganografiu a bližšie sme popísali jednu z jej metód, maticové vkladanie. Nasleduje kapitola, v ktorej pokračujeme s teoretickými znalosťami potrebnými k návrhu vlastného steganografického systému. Rozoberieme samosynchronizačné kódy a ich rozdiel so samoopravnými.

3. Samosynchronizačné kódy

Kanál, po ktorom prenášame dáta, nemusí byť stopercentný, ako uvidíme v kapitole o VoIP 4. Môže dôjsť k zmene bitu na iný, dochádza k inverzii symbolu. V tom prípade je vhodnou metódou na rekonštrukciu pôvodnej správy zvoliť samoopravný kód.

Ak dôjde k strate bitov, máme niekoľko možností na nápravu. Pokiaľ by sme vedeli, koľko bitov a na akých miestach sa stratili, tak by sme na ich miesta dali nejakú náhodnú hodnotu. Na opravu by sme mohli použiť samoopravné kódy. Často takéto informácie nemáme a so samoopravnými kódmi nevystačíme. Tento problém však riešia samosynchronizačné kódy [16, 22]. Sú vhodné nielen ak dochádza k strate bitov, ale aj ak dochádza k ich pridávaniu.

Táto kapitola nám ponúkne základnú predstavu o samosynchronizačných kódoch. Predstavíme T-kódy a popíšeme ich vlastnosti.

3.1 Úvod do samosynchronizačných kódov

Kanálom odosielame a aj prijímame prúd bitov. Bity neinterpretujeme samostatne, ale v skupine bitov, v bitových slovách. Procesu rozdelenia prúdu na slová budeme hovoriť *synchronizácia*. Presnejšie, pôjde o proces určenia začiatku a konca slov.

Prístupov k riešeniu je viac. Jedným z nich je mať pevnú dĺžku slov. Po prvotnej synchronizácii stačí odpočítať príslušný počet bitov. Problémom je efektivita z pohľadu množstva prenášaných dát. Časté slová budú mať rovnakú dĺžku ako tie menej frekventované. Ďalšou nevýhodou takejto konštrukcie je desynchronizácia dát v prípade, že dôjde k strate či pridaniu bitu.

Pre predstavu uveďme jednoduchý prípad, ako prebieha synchronizácia v bežnom texte. Slová, v tomto prípade nad nebitovou abecedou, sú rôznej dĺžky a synchronizáciu zabezpečujú medzery a interpunkcia. Vždy, keď vidíme medzeru, prípadne interpunkčné znamienko, vieme, že je koniec slova. Pre jazyk je tiež prirodzené, že často frekventované slová sú kratšie, napríklad „a, i, aj, o, od, . . . “. Medzery majú aj ďalší význam. Jednoznačne určujú slová. Napríklad bez medzier „Samozrejme“ môže byť slovo „Samozrejme“ ako aj „Samo zrejme“.

Podľa [15] je priemerná dĺžka slov v slovenskom jazyku vyjadrená v slabikách 2,3. To znamená, že približne každých $6 - 7$ znakov sa vyskytne medzera, synchronizačný znak by sa vyskytoval s frekvenciou približne 15%.

Nech máme kód s premenlivou dĺžkou slova a nech nastane chyba. Synchronizácia sa stratí, dôjde k desynchronizácii dátového toku a chybnému odkódovaniu slov.

Zaveďme značenie platné počas celej tejto kapitoly. Budeme pracovať nad *abecedou* Σ. *Množinu všetkých slov* nad abecedou ľubovoľnej dĺžky budeme označovať Σ^*. *Prázdne slovo* budeme označovať λ a *množinu neprázdnych slov* $\Sigma^+ = \Sigma^* \backslash \{\lambda\}$. Množinu $\mathcal{S} \subseteq \Sigma^*$ nazývame *jazyk*.

Definícia 10. *Nech* $\mathbf{x}, \mathbf{y} \in \Sigma^*$ *a nech* λ *je prázdne slovo. Hovoríme, že* $\mathbf{y}$ je faktor $\mathbf{x}$*, ak platí* $\mathbf{x} = \mathbf{uyv}$*, pre* $\mathbf{u}, \mathbf{v} \in \Sigma^*$*. Ak* $\mathbf{u} = \lambda$*, hovoríme, že* $\mathbf{y}$ *je* prefix $\mathbf{x}$*, píšeme* $\mathbf{y} \preceq \mathbf{x}$*. Ak* $\mathbf{v} = \lambda$*, hovoríme, že* $\mathbf{y}$ *je* sufix $\mathbf{x}$*, píšeme* $\mathbf{x} \succeq \mathbf{y}$*.*

V predchádzajúcej kapitole sme definovali blokový kód. V tejto kapitole sa budeme zaoberať kódmi s variabilnou dĺžkou slov.

Definícia 11. *Majme abecedu* Σ. *Nech* $\mathcal{S} \subseteq \Sigma^*$ *je* jazyk. *Ak je* $\mathcal{S}$ *konečný neprázdny jazyk, hovoríme o* kóde.

Počas celej kapitoly budeme uvažovať $\mathcal{S}$ konečné. Budeme sa pridržiavať rovnakej terminológie ako pri blokových kódoch a prvky Σ^* nazveme slová a prvky kódu $\mathcal{S}$ kódové slová. Na rozdiel od blokových kódov sú slová a aj kódové slová premenlivej dĺžky.

Definícia 12. *Hovoríme, že jazyk (kód)* $\mathcal{S} \subseteq \Sigma^*$ *je* bezprefixový, *ak:*

- $\lambda \notin \mathcal{S}$,
- $\forall \mathbf{x}, \mathbf{y} \in \mathcal{S} : \mathbf{x} \preceq \mathbf{y} \Rightarrow \mathbf{x} = \mathbf{y}$.

Definícia 13. *Nech* $\mathcal{S}$ *je kód a nech* $\mathbf{x}, \mathbf{y} \in \mathcal{S} \subseteq \Sigma^*$ *sú kódové slová.* Samosynchronizačný kód *je kód, pre ktorý platia nasledujúce podmienky:*

- $\forall \mathbf{x}, \mathbf{z} \in \mathcal{S}$*: ak je* $\mathbf{z}$ *faktor* $\mathbf{x}$, *tak* $\mathbf{z} = \mathbf{x}$,
- $\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{S}$*: ak je* $\mathbf{z}$ *faktor* $\mathbf{xy}$, *tak* $\mathbf{z} = \mathbf{x}$ *alebo* $\mathbf{z} = \mathbf{y}$.

3.2 T-kódy

T-kódy sú bezprefixové samosynchronizačné kódy. Ich prednosťou je, že sa vedia rýchlo zotaviť z nežiaducich vplyvov šumu a znovu zosynchronizovať dáta.

Typicky budeme pracovať s binárnou abecedou, $\Sigma = \{0, 1\}$. Treba poznamenať, že T-kódy sú štatisticky synchronizovateľné, teda synchronizácia nie je nutným predpokladom k správnemu dekódovaniu [21].

3.2.1 Konštrukcia T-kódov

Predtým, než zadefinujeme a skonštruujeme T-kódy, zadefinujme pojem *T-rozšírenie.*

Definícia 14. *Nech* $\mathcal{S} \subseteq \Sigma^+$, $\mathbf{p} \in \mathcal{S}$. T-rozšírenie *jazyku* $\mathcal{S}$ *pomocou* T-prefixu $\mathbf{p}$ *definujme ako* $\mathcal{S}_{(\mathbf{p})} = (\mathcal{S} \setminus \{\mathbf{p}\}) \cup \mathbf{p}\mathcal{S}$.

Majme $n \in \mathbb{N}^+$, rekurzívne vieme T-rozšírenie zapísať:

$$\mathcal{S}_{(\mathbf{p}_1,\ldots,\mathbf{p}_n)} = \begin{cases} (\mathcal{S})_{\mathbf{p}_1} & \text{pre } n = 1, \\ (\mathcal{S}_{(\mathbf{p}_1)})_{\mathbf{p}_2} & \text{pre } n = 2, \\ (\mathcal{S}_{(\mathbf{p}_1,\ldots,\mathbf{p}_{n-1})})_{\mathbf{p}_n} & \text{pre } n > 2. \end{cases}$$

Definícia 15. *Nech* $\mathcal{S} \subseteq \Sigma^+$, $\mathbf{p}_1 \in \mathcal{S}$ *a pre* $\forall i \in \mathbb{N} : 1 < i \leq n$ *a pevne dané* $n \in \mathbb{N}^+, n \geq 2 : \mathbf{p}_i \in \mathcal{S}_{(\mathbf{p}_1,\ldots,\mathbf{p}_{i-1})}$. *Množinu* $\mathcal{S}_{(\mathbf{p}_1,\ldots,\mathbf{p}_n)}$ *nazývame* T-kód.

T-kódy konštruujeme tak, že rekurzívne generujeme vyššie zadefinované množiny $\mathcal{S}_{(\mathbf{p}_1,\ldots,\mathbf{p}_n)}$ pre pevne volené prefixy $\mathbf{p}_i$:

- na začiatku majme inicializačnú množinu $\mathcal{S}$, typicky $\mathcal{S} = \{0,1\}$,
- pevne zvoľme $\mathbf{p}_1 \in \mathcal{S}$ a skonštruujme množinu $\mathcal{S}_{(\mathbf{p}_1)} = (\mathcal{S} \setminus \{\mathbf{p}_1\}) \cup \mathbf{p}_1\mathcal{S}$,
- $\forall i \in \mathbb{N}, i \geq 2$, pevne zvoľme prefix $\mathbf{p}_i \in \mathcal{S}_{(\mathbf{p}_1 \dots \mathbf{p}_{i-1})}$ a skonštruujme množinu $\mathcal{S}_{(\mathbf{p}_1 \dots \mathbf{p}_i)} = (\mathcal{S}_{(\mathbf{p}_1 \dots \mathbf{p}_{i-1})} \setminus \{\mathbf{p}_i\}) \cup \mathbf{p}_i\mathcal{S}$.

T-kódy môžeme reprezentovať aj pomocou stromu[1], v našej práci budeme uvažovať len binárne stromy. Konštrukcia T-kódov je v takomto prípade tvorená operáciami na tomto strome:

- na začiatku majme inicializačný strom T_0, typicky je T_0 tvorené koreňom, ktorý predstavuje prázdne slovo λ, a práve 2 listami, označme ich 0 a 1. Slová v listoch tvoria množinu $\mathcal{S} = \{0,1\}$,
- $\forall i \in \mathbb{N}, i \geq 1$: označme T_{i-1}^c kópiu stromu T_{i-1}. Strom T_i vznikne pripojením stromu T_{i-1}^c do pevne zvoleného listu stromu T_{i-1}. Označme daný list $\mathbf{p}_i$. Slová v listoch stromu tvoria množinu $\mathcal{S}_{(\mathbf{p}_1 \dots \mathbf{p}_i)}$.

Každý bezprefixový kód $\mathcal{S} \subseteq \Sigma^*$ vieme reprezentovať pomocou stromu. Ten slúži na dekódovanie a jeho listy sú kódové slová.

Majme T-kód $\mathcal{S}_{(\mathbf{p}_1 \dots \mathbf{p}_i)}, i \in \mathbb{N}^+$. Koreň stromu je prázdne slovo a listy sú prvky množiny $\mathcal{S}_{(\mathbf{p}_1 \dots \mathbf{p}_i)}, i \in \mathbb{N}^+$. Zvyšné uzly sú prvky množín $\mathcal{S}, \mathcal{S}_{(\mathbf{p}_1)}, \dots \mathcal{S}_{(\mathbf{p}_1 \dots \mathbf{p}_{i-1})}$, ktoré vznikli iteratívnym procesom popísaným vyššie.

Ak volíme dané listy (prefixy) tak, aby sme vyčerpali všetky prefixy predom danej konkrétnej dĺžky, tak hovoríme o *úplnom T-rozšírení*. Úplne T-rozšírenie popisuje obrázok 3.1.

Na pravom strome v obrázku 3.1 máme 33 listov. Inak povedané, máme 33 kódových slov T-kódu $\mathcal{S}_{(1,0,00,10,11)}$. Najkratšie je dĺžky 3, najdlhšie 9. Na množine $\mathcal{S}_{(1,0,00,10,11)}$ vytvorme vlastný kód, ako znázorňuje tabuľka 3.1. Jednotlivým kódovým slovám priraďujeme znaky, ktoré chceme zakódovať. Sú to interpunkcia, medzera, znak pre koniec riadku a písmená základnej abecedy bez diakritiky.

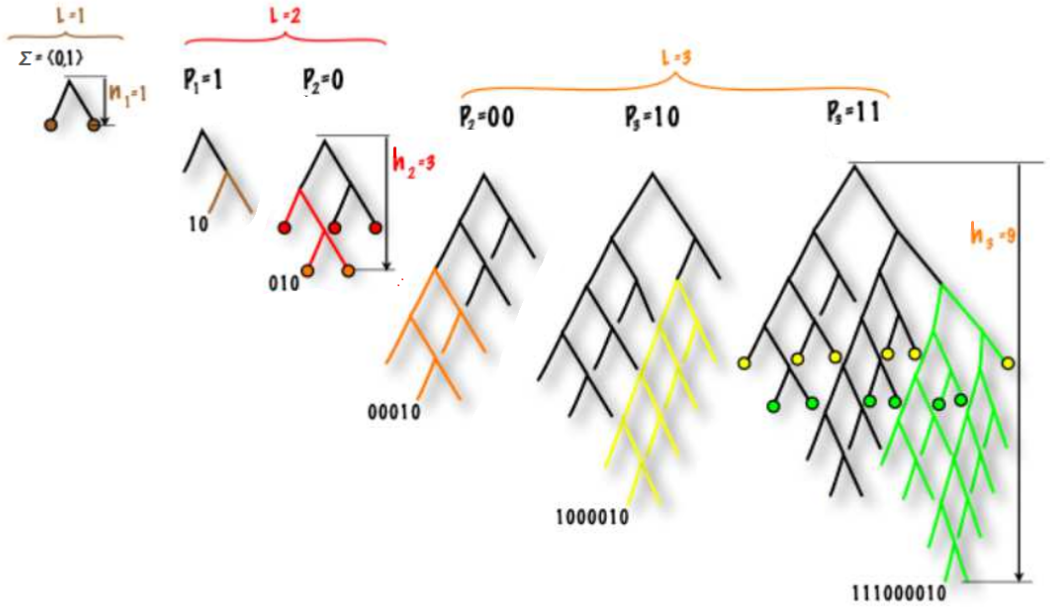
Najfrekventovanejšia je medzera, preto jej priraďme najkratšie (kódové) slovo. Frekventovaným písmenám priraďme kratšie slová, než menej frekventovaným. Frekvenčnú analýzu písmen (po odstránení diakritiky) v slovenskom jazyku vypracovali autori Štefánik, Rusko a Považanec vo svojej práci [20]. Vypracovali sme vlastný „slovník“, ktorým pomocou substitúcie kódujeme a dekódujeme. Aby kódovanie a dekódovanie mohlo prebiehať správne, predpokladajme, že odosielateľ a aj príjemca pracujú s rovnakým slovníkom. Dohodnú sa na ňom pred tým, než začne samotná komunikácia.

Predpokladajme, že nevieme, kedy začal prenos dát, ani či nastala chyba. Synchronizácia prebieha rýchlo, typicky trvá $2 - 3$ kódové slová [22], kým dôjde k zotaveniu z chyby a správnej synchronizácii.

Príklad. Zakódujme slovo „SAMOZREJME“.

S	A	M	O	Z	R	E	J	M	E
1111	011	100011	0010	110010	00010	0011	111010	100011	0011

[1] Ako pracujeme s kódmi tvorenými pomocou stromov si ukážeme neskôr. Väčšinou máme množinu slov, ktorú potrebujeme zakódovať a každému slovu z množiny priradíme práve jedno kódové slovo. Kódovanie a dekódovanie prebieha ako substitúcia.



Obr. 3.1: Úplne T-rozšírenie.
Abeceda je označená písmenom Σ. Hodnota L udáva dĺžku prefixu. Volené prefixy sú $\mathbf{p}_i, i \in \mathbb{N}, 1 \leq i \leq 5$. Hĺbku stromu v i-tom kroku vyjadruje hodnota h_i. Zdroj: [23].

č.	slovo	znak	frekvencia
1	010	␣	
2	011	A	9,57
3	0010	O	8,62
4	0011	E	8,12
5	0000	I	6,48
6	1010	N	5,65
7	1011	T	5,00
8	1111	S	4,98
9	00010	R	4,65
10	00011	V	4,20
11	10010	L	4,15
12	10011	K	3,61
13	11010	C	3,32
14	11011	U	3,26
15	100010	D	3,20
16	100011	M	2,97
17	100000	P	2,72

č.	slovo	znak	frekvencia
18	110010	Z	2,72
19	110011	Y	2,44
20	110000	H	2,26
21	111010	J	1,86
22	111011	B	1,49
23	1000010	G	0,6
24	1000011	F	0,27
25	1100010	X	0,17
26	1100011	Q	
27	1110010	W	
28	1110011	.	
29	11100010	,	
30	11100011	?	
31	11100000	!	
32	111000010	$\backslash n$	
33	111000011	;	

Tabuľka 3.1: Tabuľka vlastného kódovania abecedy pomocou T-kódu $\mathcal{S}_{(1,0,00,10,11)}$.
Znak ␣ označuje medzeru a $\backslash n$ nový riadok.

Nech nastane chyba a vypadne nám tretí bit. Dekódovanie prebehne nasledovne:

111011	100011	0010	110010	00010	0011	111010	100011	0011
B	M	O	Z	R	E	J	M	E

Vidíme, že dekódovanie sa dokázalo zotaviť po chybe. Vypadnutý bit spôsobil zmenu „SA“ na „B“ a prijali sme slovo „BMOZREJME“.

3.2.2 Vlastnosti T-kódov

Pomocou algoritmu definovaným T-rozšírením konštruujeme kód nad ľubovoľnou abecedou Σ. V našom prípade je $\Sigma = \{0, 1\}$. Všimnime si, že sa naň môžeme pozerať aj ako na algoritmus, ktorý vytvára slová určitej dĺžky. Majme T-kód $\mathcal{S}_{(\mathbf{p}_1 \dots \mathbf{p}_n)}$, $n \in \mathbb{N}^+$. Potom každé najdlhšie slovo musí prechádzať všetkými prefixmi $\mathbf{p}_1 \dots \mathbf{p}_n$, každé najdlhšie slovo je tvorené zreťazením $\mathbf{p}_n||\mathbf{p}_{n-1}|| \dots ||\mathbf{p}_1||a$, kde $a \in \mathcal{S}$ je najdlhšie slovo v $\mathcal{S}$.

Príklad. V obrázku 3.1 pripájame jednotlivé stromy postupne na prefixy $\mathbf{p}_1 = 1$, $\mathbf{p}_2 = 0$, $\mathbf{p}_3 = 00$, $\mathbf{p}_4 = 10$ a $\mathbf{p}_5 = 11$. Najdlhšie slovo je $111000010 = \mathbf{p}_5||\mathbf{p}_4||\mathbf{p}_3||\mathbf{p}_2||\mathbf{p}_1||0$.

Pre nás je najzaujímavejšou vlastnosťou synchronizácia. Vlastnosť samosynchronizácie je spôsobená konštrukciou kódu [22, 21]. Ako sme videli na poslednom príklade v predchádzajúcej sekcii, k synchronizácii dochádza samotným dekódovaním.

Zoznámili sme sa so základnými definíciami a vlastnosťami T-kódov. Pre nás budú významné len dve vlastnosti. Bezstratové komprimovanie, ktoré, ako uvidíme neskôr, bude slúžiť ku komprimácii posielanej správy a samosynchronizácia. Bezstratové komprimovanie sme docielili vhodným kódovaním znakov na jednotlivé kódové slová, pričom sme dbali na frekvenčnú analýzu písmen slovenského jazyka [20]. Docielili sme tým kódovanie častých slov na kratšie kódové slová než menej častých slov.

Touto kapitolou ukončíme teoretické poznatky, ktoré potrebujeme k návrhu nášho steganografického systému.

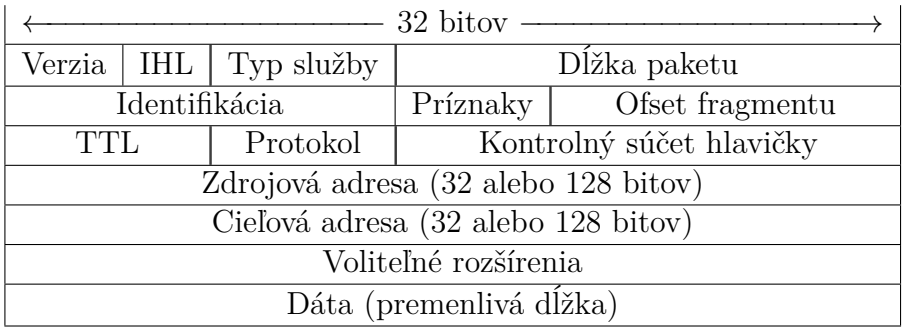
4. Voice over IP

Pri prenose dát po kanáli môže dôjsť k strate či poškodeniu dát. Pokiaľ nedôjde k náprave, prenos nazveme *nespoľahlivý*. Častokrát od kanálu očakávame, že bude *spoľahlivý*. To znamená, že kanál zabezpečí úplný a bezchybný prenos dát.

Spoľahlivý prenos nie je vždy výhodnejší, než nespoľahlivý. Náprava straty a poškodenia dát je spojená s réžiou, ktorá nemusí vyhovovať každému prenosu. Zaťažuje to výpočtovú a aj prenosovú kapacitu. Napríklad je potrebné skontrolovať bezchybnosť prijatých dát a následne poslať potvrdenie odosielateľovi, aby vedel, v akom stave a či vôbec dáta dorazili k príjemcovi. V prípade potreby odosielateľ pošle chýbajúce dáta znovu. Treba si uvedomiť, že spoľahlivosť kanálu je spojená s oneskorením prenosu a nepravidelnosťou v doručovaní. Naša práca pojednáva o prenose hlasu. Oneskorenie a hlavne nepravidelnosť v doručovaní by na hlas pôsobili príliš rušivo. Zato občasné vynechanie alebo poškodenie dát nemusíme ani postrehnúť.

V tejto kapitole sa budeme zaoberať s Voice over IP a protokolmi IP a UDP [4]. Následne popíšeme Skype pakety [2].

Voice over IP, skrátene VoIP, zaisťuje prenos hlasu nad prenosovým protokolom IP (z anglického Internet Protocol). Výhodou je, že IP vytvára jednotné prostredie pre všetky aplikácie. Posielané dáta delíme na bloky, ktoré sú vkladané do tzv. paketov. IP pracuje s premenlivou veľkosťou paketu, nazývaného aj IP datagram. Na obrázku 4.1 sú znázornené jednotlivé časti IP paketu.



Obr. 4.1: IP paket.

IP paket je zložený z hlavičky a samotných dát. Hlavička je tvorená nasledujúcimi poliami:

- Verzia (Version) vyjadruje, či sa jedná o použitie verzie IPv4 alebo IPv6. Hlavným rozdielom medzi nimi je veľkosť adresy. IPv4 používa 32 bitové adresy a IPv6 128 bitové adresy.
- IHL (IP Header Length) určuje dĺžku hlavičky.
- Typ služby (Type of Service) popisuje zachádzanie s paketom. Pakety môžu mať priradenú rôznu kvalitu služieb.
- Dĺžka paketu (Total Length) je dĺžka celého paketu v bytoch.
- Identifikácia (Identification) je číslo jednoznačne určujúce skupinu fragmentov patriacu jednému paketu.

- Príznaky (Flags) sú 3 bitové číslo. Príznaky špecifikujú, či je možné paket fragmentovať . To znamená rozdeliť ho na menšie časti. Pokiaľ bol paket už fragmentovaný, tak pomocou bitov vieme určiť, či je daný paket posledným z fragmentovaných paketov, alebo či za ním ešte nejaké nasledujú.
- Ofset fragmentu (Fragment Offset) identifikuje pozíciu fragmentu v originálnom pakete.
- TTL (Time To Live) je číslo, ktoré vyjadruje, koľkokrát je možné paket presmerovať pred tým, než ho zničíme. Tým zabraňujeme jeho zacykleniu v sieti.
- Protokol (Protocol) identifikuje protokol, ktorým paket následne spracovávame. V našom prípade to bude UDP. Tento protokol ešte v tejto kapitole popíšeme.
- Kontrolný súčet hlavičky (Header checksum) overuje, či prenosom nedošlo k poškodeniu hlavičky.
- Zdrojová adresa (Source address) je IP adresa odosielateľa.
- Cieľová adresa (Destination address) je IP adresa príjemcu.
- Voliteľné rozšírenia (Options) voliteľné pole, v ktorom riešime napríklad bezpečnosť.

Pre lepšiu orientáciu píšeme v zátvorkách anglický ekvivalent názvu. Popis je stručný a slúži na prehľad toho, aké rôzne informácie okrem dát v paketoch prenášame. Podrobnejšie informácie nájdeme v [4].

IP je nespojový a nespoľahlivý protokol. Nespoľahlivosť sme už popísali vyššie. Chyba a ani strata dát sa nijako neošetrujú. Nespojový protokol znamená, že medzi komunikujúcimi stranami nezakladáme spojenie. Dokonca ani neoverujeme existenciu príjemcu. Každý paket je označený svojim príjemcom a cestuje po kanáli nezávisle na ostatných paketoch. To vedie k tomu, že každý paket môže cestovať inou cestou a to spôsobí, že dáta môžu prísť v rôznom poradí.

Nad IP sú dva základné typy protokolov, UDP (z anglického User Datagram Protocol) a TCP (z anglického Transmission Control Protocol). Nebudeme sa podrobnejšie zaoberať ich rozdielom. Vo všeobecnosti platí, že TCP volíme, ak od prenosu vyžadujeme, aby bol spoľahlivý. V našom prípade budeme pracovať s UDP protokolom.

UDP je rovnako ako IP nespoľahlivý a nespojový protokol. UDP paket je zložený z hlavičky a dát a je vložený (zabalený) do IP paketu. Teda UDP paket tvorí IP dáta. Hlavička UDP paketu obsahuje štyri časti: zdrojový a cieľový port, dĺžku hlavičky spolu s dátami a kontrolný súčet. Kontrolný súčet je voliteľná položka a zaručuje integritu dát. Kontrolný súčet počítame z celého UDP paketu spolu so pseudohlavičkou. Pseudohlavičku neposielame, obsahuje IP adresy odosielateľa a aj príjemcu, identifikátor protokolu a dĺžku UDP paketu.

Vráťme sa späť k VoIP. Má rôzne výhody a nevýhody. Jednu z výhod sme popísali, pracuje nad IP. Pre dátový a aj hlasový prenos využívame rovnakú technológiu. Ďalšou výhodou je cena. Má však aj nevýhody. Prenos sa snažíme mať

Zdrojový port	Cieľový port
Dĺžka	Kontrolný súčet
Dáta	

Obr. 4.2: UDP paket.

čo najmenej oneskorený a musíme dať pozor na pravidelnosť prijímania paketov. Dochádza k strate paketov, je potrebné hlas komprimovať a digitálne kódovať.

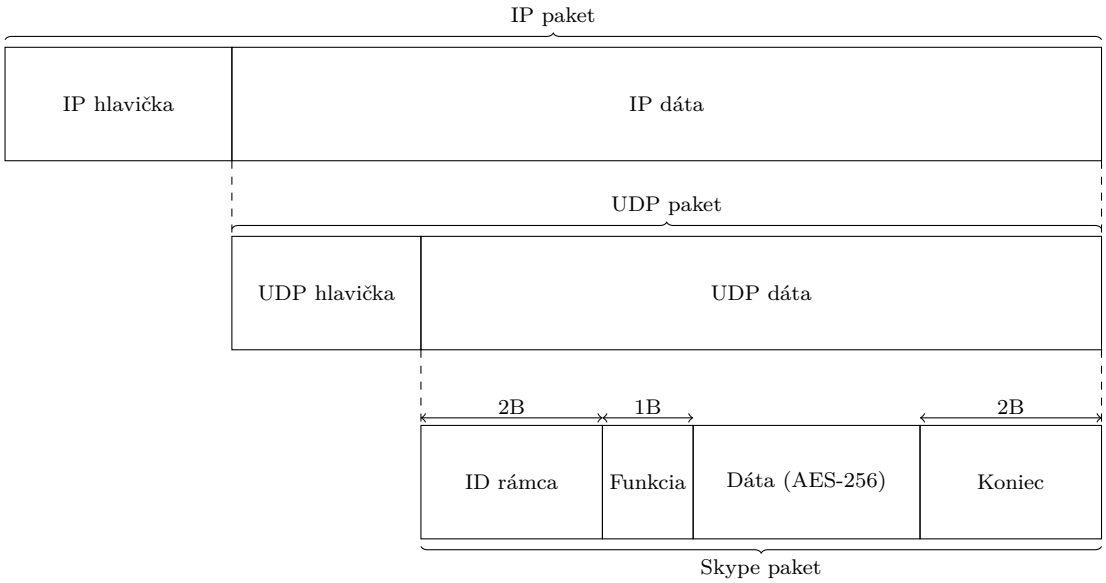
Spomeňme pár postrehov, ktoré je dobré si uvedomiť. K oneskoreniu nedochádza len pri réžii spojenej so spoľahlivým prenosom. Pripomeňme, že VoIP pracuje nad UDP protokolom, a teda nespoľahlivým prenosom. Za oneskorenie pokladáme čas, ktorý prejde od vyslania na jednej strane až k príjmu na druhej strane. Vplýva na to veľa faktorov, od rýchlosti šírenia signálu, po komprimovanie, preposielavanie paketov a ich presúvanie do front a podobne.

V sieti dochádza k strate paketov. Tá je bežná a dokonca aj očakávaná. Veľa protokolov pracuje so stratou paketov a tak zisťuje stav siete. Prípadne sa protokoly pri zlom stave prispôsobia a znížia počet posielaných paketov.

VoIP tiež zisťuje, kedy dochádza k hlasovej aktivite. Hlas meria v decibeloch a na základe veľkosti rozhoduje, kde je vhodné hlas rozdeliť pri vkladaní do jednotlivých paketov. Pri poklese amplitúdy počká predom stanovený čas, typicky 200 ms, a potom prestane vkladať hlas do paketu. Problémom je správne určenie začiatku a konca reči.

4.1 Skype

Známym príkladom VoIP je Skype. Komunikácia prebieha pomocou už spomínaných UDP paketov a dáta vkladáme do tzv. Skype paketov, obrázok 4.3. Podrobné fungovanie programu Skype je popísané v práci [2]. Číselné dáta sú



Obr. 4.3: Skype paket.

v paketoch zapisované v tzv. big alebo little endian.

Endianita vyjadruje zápis (uloženie) čísel v pamäti počítača. Definuje, v akom poradí sa uložia jednotlivé byty číselného dátového typu. Majme byty $\mathbf{n}_1, \ldots, \mathbf{n}_n$, kde $n \in \mathbb{N}^+$, pričom $\mathbf{n}_1$ je najmenej významný byte, teda byte obsahujúci najmenej významný bit, a $\mathbf{n}_n$ je najvýznamnejší byte, byte obsahujúci najvýznamnejší bit. Sú dva základné spôsoby ukladania čísel v pamäti:

- *little endian*: najmenej významný byte sa uloží do pamäte na miesto s najnižšou adresou (prvý) a za ním nasledujú ostatné, byty sú uložené v poradí $\mathbf{n}_1, \ldots, \mathbf{n}_n$,
- *big endian*: na pamäťové miesto s najnižšou adresou uložíme najviac významný byte, byty sú uložené v poradí $\mathbf{n}_n, \ldots, \mathbf{n}_1$. Tento spôsob je typický pre internetové protokoly (IPv4, IPv6, UDP, TCP, ...).

Hlas je v Skype paketoch šifrovaný pomocou šifry AES. AES [3] (z anglického Advanced Encryption Standard) je šifrovací štandard založený na symetrickej blokovej šifre. Veľkosť bloku je pevne daná na 128 bitov. Dáta šifrujeme pomocou verzie AES-256, kľúč je dlhý 256 bitov. Základom sú permutácie a substitúcie, ktoré prebiehajú v 14-tich kolách. Skype využíva vlastný šifrovací mód, ktorý je variáciou counter módu [2]. Zmena jedného bitu v šifrovaných dátach znamená zmenu jedného bitu v otvorených dátach.

Nám postačí vedieť, že zatiaľ nie je známy žiaden netriviálny kryptoanalytický útok, ktorý by prelomil AES s plným počtom kôl.

V Skype paketoch používame kontrolný súčet CRC-32. CRC (z anglického Cyclic Redundancy Check) dostane na vstupe binárne dáta $b_1, b_2, \ldots, b_n$. Dáta interpretujeme ako polynóm $\mathrm{f}(x) = b_1 x^{n-1} + b_2 x^{n-2} + \cdots + b_{n-1}x + b_n$, $n \in \mathbb{N}$. Počítame $\mathrm{f}(x)$ modulo $\mathrm{g}(x)$ v okruhu $\mathbb{F}_2[x]$, kde $\mathrm{g}(x)$ je pre CRC-32: $\mathrm{g}(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$.

Popíšme bližšie jednotlivé zložky Skype paketu (rámca):

- ID rámca je 2 bytové číslo zapísané v big endian. Identifikuje daný paket v rámci dátového prúdu a je vygenerovaný generátorom 1. Neskôr ukážeme aj jeho druhý význam, dá sa z neho odvodiť identifikátor dátového prúdu[1].
- Funkcia je 1 bytové číslo, ktorého najnižšie (najmenej významné) 4 bity vyjadrujú typ paketu. Pre textovú a audio komunikáciu je to 13_{10}. Zvyšné 4 bity sú doplnené lineárnym kongruentným generátorom, viď algoritmus 1 (zdroj [2]).
- Dáta nemajú pevnú dĺžku. Sú šifrované pomocou šifry AES-256 s variáciou counter módu.
- Koniec je 2 bytové číslo zapísané v little endian. Ako uvidíme neskôr, slúži k odvodeniu poradového čísla Skype paketu.

Jednotlivé zložky paketu slúžia na kontrolu jeho správnosti. Počítame dva kontrolné súčty. Jeden je počítaný zo zašifrovaných dát spolu s koncom. Ďalší len

[1]Identifikátor dátového prúdu určuje danú komunikáciu medzi komunikujúcimi stranami pri Skype hovore.

Algoritmus 1 Generátor

Vstup: `stav`,
rozsah generovaného čísla m
Výstup: `stav`, `rnd`
`stav` = 69069`stav` + 17009 mod 2^{32}
`rnd` = `stav` mod m
return `stav`, `rnd`

zo zašifrovaných dát:

$$\text{crc}_1 := \text{CRC-32(šifrované dáta } || \text{ koniec)},$$
$$\text{crc}_2 := \text{CRC-32(šifrované dáta)}.$$

Pozíciu paketu v dátovom prúde získame ako

$$\text{poradové\_číslo:= } (\text{crc}_2 \oplus \text{koniec}) \wedge \text{FFFF}_{16}.$$

Označme

$$\text{tag} := (\text{crc}_1 \oplus \text{ID\_rámca}) \wedge \text{FFFF}_{16}.$$

Pre audio dáta je tag nepárne číslo.

$$\text{ID\_prúdu} := \text{tag} \gg 1$$

je identifikátor dátového prúdu, kde $\gg$ je bitový posun tagu o 1 doprava a $\wedge$ je bitový AND.

V tejto kapitole sme popísali, ako vyzerajú pakety, do ktorý budeme implementovať posielanie tajnej správy. Musíme pri tom dať pozor, aby sme zachovali integritu nielen Skype paketov, ale aj UDP a IP paketov. Nasledujúca kapitola sa bude zaoberať konkrétnym návrhom steganografického modelu pre posielanie správ v Skype paketoch.

5. Model a implementácia

K skrývaniu správy do VoIP môžeme pristupovať viacerými spôsobmi. Niektoré sme načrtli v kapitole 1. V tejto kapitole rozoberieme posielanie tajných správ v Skype paketoch. Navrhneme steganografickú metódu vkladania a extrakcie správ a naimplementujeme ju pre Skype hovory.

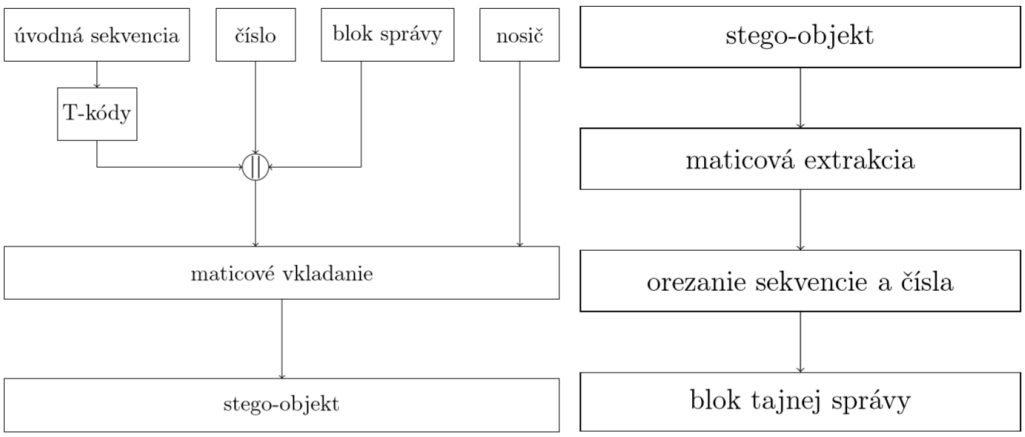
Budeme pri tom uvažovať pasívneho útočníka. Navrhneme steganografický systém, pri ktorom odpočúvaním komunikácie nebude útočník schopný odhaliť použitie steganografie. Čiastočne je model odolný aj voči aktívnemu útočníkovi, ktorý by zahadzoval pakety. Odolnosť voči takémuto útočníkovi musíme brať do úvahy, pretože pracujeme s nespoľahlivým kanálom.

Tajnú správu, ktorú budeme posielať, označme $\mathbf{m} \in \{0,1\}^l$, $l \in \mathbb{N}$. Nosičom správy sú zašifrované dáta v Skype paketoch. Zatiaľ nebudeme uvažovať dĺžku správy a nosiča. Uvažujme paket ako sekvenciu binárnych dát.

Prvá časť kapitoly bude obsahovať model navrhnutého steganografického systému. Popíšeme úvahy o tom, ako najvhodnejšie vkladať správu do Skype paketov a na čo dať pozor. Druhá časť bude nasledovať implementáciou a popisom algoritmov v pseudokóde.

5.1 Model

Na obrázku 5.1 je stručný popis navrhnutého modelu steganografického systému. Nosičom správ sú zašifrované dáta v Skype paketoch. Tajnou správou je text zložený z interpunkcie, medzery, znaku pre koniec riadku a písmen základnej abecedy bez diakritiky (ako v tabuľke 3.1). Text pred vkladaním do paketov zakódujeme pomocou T-kódov a Hammingových kódov. Potom ho rozdelíme do blokov správ. Každý blok správy vkladáme práve do jedného paketu (do jeho zašifrovaných dát). Ako delíme správu do blokov a ako vyberáme pakety, do ktorých vložíme správu, ukážeme neskôr. Na obrázku 5.1 vidíme základnú štruktúru steganografického modelu pri vkladaní jedného bloku správy.



(a) Vkladanie bloku správy. (b) Extrahovanie bloku správy.

Obr. 5.1: Steganografický systém modelu.

Popíšme úvahy, ktoré viedli k tomuto modelu systému a ukážme jeho výhody. Začnime diskusiou o vkladaní správy do jedného paketu.

5.1.1 Vkladanie do jedného paketu

Tajnú správu budeme vkladať do Skype paketu. Uvážme, ktorá časť Skype paketu bude na to najvhodnejšia. Pripomeňme jeho štruktúru (podrobnejšie v podkapitole 4.1) a ako prebieha zachovanie jeho integrity.

Každý Skype paket obsahuje časti: ID rámca, funkcia, zašifrované dáta a koniec. Overenie integrity prebieha nasledovne:

$$\begin{aligned} \text{crc}_1 &:= \text{CRC-32}(\text{dáta} \parallel \text{koniec}), && (5.1)\\ \text{crc}_2 &:= \text{CRC-32}(\text{dáta}), && (5.2)\\ \text{poradové\_číslo} &:= \text{crc}_2 \oplus \text{koniec} \wedge \text{FFFF}_{16}, && (5.3)\\ \text{tag} &:= \text{crc}_1 \oplus \text{ID\_rámca} \wedge \text{FFFF}_{16}, && (5.4)\\ \text{ID\_prúdu} &:= \text{tag} \gg 1. && (5.5) \end{aligned}$$

Musíme meniť paket tak, aby jeho poradové číslo a identifikátor prúdu, teda zo vzťahu (5.5) aj tag, zostali nezmenené.

Začnime uvažovať vkladanie správy do časti koniec. Objekt, ktorý vznikne vložením správy do časti koniec, nazvime nový_koniec. Aby bolo poradie paketu zachované, musí platiť:

$$\text{nové\_crc}_2 = \text{poradové\_číslo} \ \oplus \ \text{nový\_koniec} \wedge \text{FFFF}_{16}.$$

To znamená, že musíme zmeniť dáta na nové_dáta tak, aby platilo:

$$\text{nové\_crc}_2 = \text{CRC-32}(\text{nové\_dáta}).$$

Nevieme vložiť správu do konca bez toho, aby sme nezmenili aj šifrované dáta.

Ak by sme zvolili vkladanie správy do časti ID_rámca, vznikne objekt, ktorý nazveme nové_ID_rámca. Tag je pevne daný.

$$\text{Nové\_crc}_1 = \text{nové\_ID\_rámca} \ \oplus \ \text{tag} \wedge \text{FFFF}_{16}.$$

To znamená, že musíme zmeniť dáta a koniec na nové_dáta a nový_koniec tak, aby platilo:

$$\text{nové\_crc}_1 = \text{CRC-32}(\text{nové\_dáta} \parallel \text{nový\_koniec}).$$

Nevieme vložiť správu do ID_rámca bez toho, aby sme nezmenili aj šifrované dáta.

V oboch prípadoch prenášame dáta len v dvoch bytoch za cenu zmeny dát, ktoré prenášajú hlas. Zmenou konca a ID rámca nedokážeme zachovať poradové číslo paketu a identifikátoru prúdu bez toho, aby sme nepoškodili zvuk. Jediným prístupom, ako zachovať hlasové dáta nezmenené, je vkladať informáciu do poľa funkcie. To je veľkosti jedného bytu. Najmenej významné štyri bity musia zostať nezmenené. Vyjadrujú, o aký druh paketu ide, v našom prípade to sú audio pakety a funkcia má hodnotu 13_{10}. Teda na vkladanie tajnej informácie nám zostávajú štyri bity. Vkladanie správy do štyroch bitov na paket je neefektívne.

Meňme samotné šifrované dáta. A to tak, aby nebolo poznať prenos správy. K tomu využijeme existujúcu metódu, maticové vkladanie. Tak dokážeme naraz preniesť väčšie množstvo tajnej informácie než vkladaním do poľa funkcie, dáta sú veľkosti niekoľko desiatok až stoviek bytov.

Dáta v paketoch sú zašifrované. Teoreticky by sme mohli meniť aj väčšie množstvo bitov v pakete. My však chceme zachovať čo najvernejšiu podobnosť k pôvodnému paketu.

Budeme meniť dáta tak, aby sme zachovali poradové číslo paketu a identifikátor dátového prúdu:

- vložme správu do dát: nové_dáta := Emb(dáta, $\mathbf{m}$), kde Emb označuje maticové vkladanie,
- z (5.2): nové_crc$_2$:= CRC-32(nové_dáta),
- z (5.3): nový_koniec := (nové_crc$_2$ $\oplus$ poradové_číslo) $\wedge$ FFFF$_{16}$,
- z (5.1): nové_crc$_1$:= CRC-32(nové_dáta || nový_koniec),
- z (5.4): nové_ID_rámca := (nové_crc$_1$ $\oplus$ tag) $\wedge$ FFFF$_{16}$.

Paket je po zmene dát zložený z polí nové_ID_rámca, funkcia, nové_ dáta a nový_koniec. Tým zachováme správnosť Skype paketu. Integritu IP a UDP paketov zachováme prepočítaním kontrolných súčtov v ich hlavičke. Uvedomme si, že nie je potrebné zachovať dĺžku paketu. V prípade jej zmeny prepočítame aj dĺžky IP a UDP paketov a paket odošleme s novými údajmi. Takýto prístup by mohol byť nápadnejší, preto budeme udržiavať dĺžku paketu zhodnú s originálnym paketom.

5.1.2 Vkladanie do viacerých paketov

Špecifické pre našu prácu je, že nemáme pevne danú dĺžku nosiča. Správy vkladáme do niekoľkých paketov. Hovor na Skype môže prebiehať dovtedy, kým nepošleme celú správu. Samozrejme, musíme dať pozor, aby sme nevzbudili podozrenie príliš dlhým hovorom. Nech $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_p$, $p \in \mathbb{N}$, sú bloky správy $\mathbf{m}$. Zatiaľ uvažujme, že správa nemusí byť rozdelená do blokov rovnakej dĺžky. Potom miesto vkladania správy $\mathbf{m}$ do paketu, budeme postupne vkladať jednotlivé bloky $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_p$ do viacerých paketov.

Vďaka kontrolným súčtom predpokladáme, že každý prijatý paket dorazil v pôvodnom stave, teda v ňom nenastala chyba. Uvedomme si, že meníme dáta, ktoré sú zašifrované šifrou AES v counter móde. Zmena jedného bitu v zašifrovaných dátach spraví zmenu jedného bitu v otvorených dátach. Správu nebudeme ukrývať do každého paketu. Jeden z faktorov výberu bude vyhovujúca veľkosť paketu. Voľbu paketov ukážeme neskôr. Najprv sa venujme tomu, ako zistíme, ktoré pakety ukrývajú informáciu a ktoré nie.

Jeden spôsob je vkladanie správy do každého i-teho paketu, kde $i \in \mathbb{N}$. Nevýhoda takejto konštrukcie je, že ak dôjde k strate paketu, tak budeme hľadať správu v nesprávnych paketoch. Nastane strata tajnej informácie. Ďalší spôsob je, že dané pakety označíme. My sme zvolili tento spôsob a šifrované dáta v paketoch so správou sme označili úvodnou sekvenciou bitov, označme ju `seq`. Ak by

boli prvé bity paketov zhodné, vzbudilo by to podozrenie. Preto túto sekvenciu pripojíme k bloku správy, vznikne $\texttt{seq}||\mathbf{m}_i$, $1 \leq i \leq p$ a spolu ich maticovo vkladáme do paketov. Tým ich odlíšime od paketov, ktoré tajnú informáciu neobsahujú. Bezpečnejšie je nevoliť pevnú sekvenciu, aby nevzbudila podozrenie u pasívneho útočníka. Generujme úvodnú sekvenciu pomocou funkcie `f`, ktorej vstupom bude tajný kľúč $\mathbf{k} \in \mathcal{K}$ a poradové číslo paketu poradové_číslo. Výstupom bude úvodná sekvencia závislá na poradovom čísle paketu: $\texttt{f}(\mathbf{k},\text{poradové\_číslo}) = \texttt{seq}$. Samotný generátor `f` aj kľúč $\mathbf{k} \in \mathcal{K}$ sú známe len odosielateľovi a príjemcovi tajnej správy. Príkladom generátoru môže byť napríklad hašovacia funkcia, ktorá na vstupe dostane zreťazené slovo $\mathbf{k} \parallel$ poradové_číslo.

Nezabudnime, že pracujeme s UDP paketmi, teda nemáme zaručené správne poradie doručenia. To pre nás znamená, že musíme pakety číslovať, aby sme po ich prijatí vedeli zrekonštruovať správne poradie paketov a extrahovať z nich správu. Nie je vhodné použiť už existujúce poradové čísla paketov. Vďaka nim budeme schopní správne zoradiť pakety nesúce tajnú informáciu, ale z toho nespoznáme, či nejaké pakety chýbajú. Inak povedané, budeme vidieť, že nastala strata paketov, ale nebudeme vedieť určiť, či stratené pakety niesli správu. Tento problém by čiastočne riešili samosynchronizačné kódy [10]. My sa však snažíme stratiť čo najmenej informácie. Preto pakety nesúce správu číslujme.

Rovnakou úvahou, ako pri označení paketov, vkladajme do paketov miesto bloku správy $\mathbf{m}_i$ správu $\texttt{seq}||\texttt{num}||\mathbf{m}_i$, $1 \leq i \leq p$ a `num` je číslo paketu s uloženou správou.

Pripomeňme, že pakety nie sú rovnakej dĺžky. Vďaka číslovaniu vieme zistiť, na akom mieste je stratený paket, ale nevieme, o koľko informácie sme tým prišli. Budeme deliť správu $\mathbf{m}$ na bloky rovnakej dĺžky. Teda do paketov budeme vkladať vždy pevný počet bitov informácie. Tento postup nám umožní zrekonštruovať správu zo stratených paketov za použitia samoopravných kódov.

Ošetrenie straty paketov

Nech je správa zložená zo sekvencie bitov, $\mathbf{m} = (b_1, b_2, \ldots, b_l) \in \{0,1\}^l$, $l \in \mathbb{N}$. Rozdeľme správu do p blokov veľkosti r, r,$p \in \mathbb{N}, rp = l$ (+ prípadný padding). A to tak, že ju rozdelíme do „stĺpcov“. Teda prvý blok bude tvorený bitmi $\mathbf{m}_1 = (b_1, b_{p+1}, b_{2p+1}, \ldots, b_{(r-1)p+1})$, druhý blok bude obsahovať bity $\mathbf{m}_2 = (b_2, b_{p+2}, b_{2p+2}, \ldots, b_{(r-1)p+2})$ a tak ďalej. Posledný blok bude tvorený bitmi $\mathbf{m}_p = (b_p, b_{2p}, b_{3p}, \ldots, b_l)$. Týmto rozdelením do blokov pri strate paketu nestratíme bity pri sebe. Na ich opravu budeme môcť použiť samoopravné kódy. Konkrétne budeme uvažovať Hammingove kódy. Tie dokážu opraviť len jednu chybu.

Nech p je počet všetkých paketov, potom dokážeme opraviť stratené pakety, ktoré budú od seba aspoň o dĺžku kódu. Ak stratíme pakety idúce za sebou, Hammingove kódy nebudú postačujúce. Samosynchronizačné kódy, konkrétne T-kódy, zaručia rýchle zotavenie z chyby. Stratíme však časť informácií, ktoré si vyžiada správna synchronizácia. Keďže p je počet paketov, v ktorých je ukrytá správa, tak stratené informácie budú roztrúsené po celej posielanej správe. Ak by sme správu vkladali do „stĺpcov“ postupne vždy po n paketoch, n je dĺžka Hammingovho kódu, tak stratené informácie budú roztrúsené len v nejakej časti posielanej správy. Nech r je veľkosť jedného bloku a n je dĺžka kódu. Potom prvých rn bitov správy $\mathbf{m}$ rozdelíme do prvých n blokov dĺžky r. Nasledujúcich rn bitov správy do ďalších n blokov a tak ďalej. Dĺžka l musí byť násobkom rn. Ak nie je,

správa sa doplní nulami. Konštrukciu popisuje obrázok 5.2. Keď budeme hovoriť o správe v stĺpcoch, budeme mať na mysli túto konštrukciu. Ak stratíme jeden

$$
\begin{array}{c|cccccc}
 & \longleftarrow & & r & & \longrightarrow & \\
\uparrow & b_1 & b_{n+1} & b_{2n+1} & \cdots & b_{(r-1)n+1} & \rightarrow \text{ blok } \mathbf{m}_1 \\
n & b_2 & b_{n+2} & b_{2n+2} & \cdots & b_{(r-1)n+2} & \rightarrow \text{ blok } \mathbf{m}_2 \\
\downarrow & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
 & b_n & b_{2n} & b_{3n} & \cdots & b_{rn} & \rightarrow \text{ blok } \mathbf{m}_n \\
 & & & & & & \\
\uparrow & b_{rn+1} & b_{(r+1)n+1} & b_{(r+2)n+1} & \cdots & b_{(2r-1)n+1} & \rightarrow \text{ blok } \mathbf{m}_{n+1} \\
n & b_{rn+2} & b_{(r+1)n+2} & b_{(r+2)n+2} & \cdots & b_{(2r-1)n+2} & \rightarrow \text{ blok } \mathbf{m}_{n+2} \\
\downarrow & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
 & b_{(r+1)n} & b_{(r+2)n} & b_{(r+3)n} & \cdots & b_{2rn} & \rightarrow \text{ blok } \mathbf{m}_{2n} \\
 & & & & & & \\
\uparrow & b_{(i-1)rn+1} & b_{((i-1)r+1)n+1} & b_{((i-1)r+2)n+1} & \cdots & b_{(ir-1)n+1} & \rightarrow \text{ blok } \mathbf{m}_{(i-1)n+1} \\
n & b_{(i-1)rn+2} & b_{((i-1)r+1)n+2} & b_{((i-1)r+2)n+2} & \cdots & b_{(ir-1)n+2} & \rightarrow \text{ blok } \mathbf{m}_{(i-1)n+2} \\
\downarrow & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
 & b_{((i-1)r+1)n} & b_{((i-1)r+2)n} & b_{((i-1)r+3)n} & \cdots & b_{irn} & \rightarrow \text{ blok } \mathbf{m}_{in}
\end{array}
$$

Obr. 5.2: Spracovanie správy do stĺpcov.

paket zo skupiny paketov (blokov) $\mathbf{m}_{tn+1}, \mathbf{m}_{tn+2}, \cdots, \mathbf{m}_{(t+1)n}$, $t \in \mathbb{N}$, dokážeme chybu opraviť pomocou samoopravných kódov.

Spracovanie správy do stĺpcov môžeme vylepšiť ľubovolnou inou permutáciou bitov tajnej správy. Permutácia musí byť známa len odosielateľovi a príjemcovi ukrytej správy. Pri voľbe permutácie musíme dbať na to, aby sme pri strate paketu dokázali zrekonštruovať správu Hammingovými (n,k)-kódmi. Teda hľadáme takú permutáciu, ktorá spermutuje bity tak, aby blok dát vkladaný do jedného paketu neobsahoval bity $b_{1+i}, \ldots, b_{n+i}$, kde i je násobok n. Rovnakú úvahu môžeme použiť aj na bloky dát idúcich za sebou. Chceme byť schopní zrekonštruovať dáta v prípade straty paketov idúcich za sebou. Tým sa v našom prípade nemusíme zaoberať, pretože ako popíšeme neskôr, bloky dát nebudeme vkladať do paketov idúcich bezprostredne za sebou.

Uvedieme príklad stratovosti paketov počas hovoru Skype. Pomocou okna *technical call info* sme v Skype odpozorovali, že pri dvojminútovom hovore nastala strata troch paketov. Stratovosť je nízka, ale straty paketov nastávajú pri sebe. Vplýva na to veľa faktorov, ako napríklad vyťaženosť siete, kvalita internetového pripojenia a podobne.

Týmto by sme mohli diskusiu o vkladaní správ do Skype paketov uzavrieť a popísať konkrétnu implementáciu navrhnutého steganografického nástroja.

5.2 Implementácia

V tejto časti popíšeme pseudokódmi jednotlivé algoritmy. Predtým, než popíšeme vkladanie správ do Skype hovoru na jednej strane a extrahovanie správ zo Skype hovoru na druhej strane, tak začneme popisom potrebných algoritmov.

5.2.1 Hammingove kódovanie

Ako prvé popíšme kódovanie (algoritmus 2) a dekódovanie (algoritmus 3) pomocou Hammingovho (n,k)-kódu. Popis je v pseudokódoch. Oba algoritmy náj-

Algoritmus 2 Kódovanie pomocou Hammingovho kódu

Vstup: generujúca matica G v systematickom tvare,
vstupné dáta `stream`,
dĺžka slova `len(word)`
Výstup: zakódované dáta pomocou Hammingovho kódu `enc_stream`

```
if len(stream) mod len(word) != 0 then
  doplň stream nulami (padding)
end if
enc_stream =
while v dátach stream existujú nezakódované slová do
  vezmi nasledujúce nezakódované slovo word
  vypočítaj z neho kódové slovo enc_word = word.G
  enc_stream += enc_word
end while
return enc_stream
```

deme v skripte *HammingEncoding.py* a pracujeme s binárnym Hammingovým (7,4)-kódom.

Algoritmus 3 Dekódovanie pomocou Hammingovho kódu

Vstup: kontrolná matica H,
vstupné dáta `stream`,
dĺžka slova `len(dec_word)`
Výstup: dekódované dáta pomocou Hammingovho kódu `dec_stream`

```
dec_stream =
while v dátach stream existujú neodkódované slová do
  vezmi nasledujúce neodkódované slovo code_word
  spočítaj syndróm syndrome = H.code_word^T
  nájdi i také, že i-ty stĺpec H je rovný syndrómu syndrome
  if také i existuje then
    code_word[i] ⊕= 1
  end if
  dec_word = code_word[0 : len(dec_word)]
  dec_stream += dec_word
end while
return dec_stream
```

5.2.2 Kódovanie pomocou T-kódov

Pomocou pseudokódov popíšme kódovanie (algoritmus 4) a dekódovanie (algoritmus 5)T-kódov. Budeme uvažovať rovnakú abecedu a slovník, ako v tabuľke 3.1 z kapitoly o samosynchronizačných kódoch 3. Dekódujeme, pokiaľ nenačítame

Algoritmus 4 Kódovanie pomocou T-kódu

Vstup: slovník `vocabulary`,
vstupné dáta `text`
Výstup: zakódované dáta pomocou T-kódu `enc_text`

```
enc_text =
for letter in text do
  enc_letter = vocabulary[letter]
  enc_text += enc_letter
end for
return enc_text
```

ukončovací znak, v našom prípade „;“. Slúži k tomu, aby sme vedeli, kde končia užitočné dáta a oddelili ich tak od vzniknutého paddingu. Ukončovací znak bude predom dohodnutý komunikujúcimi stranami. Alternatívnym prístupom by mohlo byť zakódovanie veľkosti správy do paketu. Tento prístup však vyžaduje ďalšiu úvahu, ako to ošetriť voči strate paketom. Tomu sa v našej práci nebudeme venovať.

Kódovanie aj dekódovanie pomocou T-kódov nájdeme v skripte *TcodeEncoding.py*.

Algoritmus 5 Dekódovanie pomocou T-kódu

Vstup: slovník `vocabulary`,
vstupné dáta `stream`
Výstup: dekódované dáta pomocou T-kódu `dec_stream`

```
dec_stream =
while len(stream) > 0 do
  code_word += stream[0]
  if code_word in vocabulary then
    letter, pre ktoré vocabulary[letter] = code_word
    if letter = ';' then
      break
    end if
    dec_stream += letter
  else
    if len(code_word) > 9 (príliš dlhé kódové slovo) then
      code_word =  (zahoď ho)
    end if
  end if
  stream = stream[1 :] (odrezanie prvého bitu)
end while
return dec_stream
```

5.2.3 Maticové vkladanie

Upravené správy, s pridanou začiatočnou sekvenciou a číslovaním, vkladáme do zašifrovaných dát pomocou maticového vkladania. Budeme pri tom využívať

binárne Hammingove kódy, konkrétne (7,4)-kód. Popis maticového vkladania sme rozobrali v kapitole o steganografii. Popíšme maticové vkladanie (algoritmus 6) a extrakciu (algoritmus 7) v pseudokóde. Implementáciu maticového vkladania

Algoritmus 6 Maticové vkladanie

Vstup: kontrolná matica H Hammingovho kódu (n,k),
vstupné dáta `stream`,
tajná správa `message`
Výstup: vložená správa `message` do `stream`: `stego_stream`
if `len(message)` mod $(n-k)$! $= 0$ **then**
doplň `message` nulami (padding)
end if
`stego_stream` =
while v `stream` existuje n-tica, do ktorej nebola vložená správa **do**
vezmi nasledujúcu n-ticu `stream_part`
spočítaj syndróm `syndrome` = H.`stream_part`^{T}
vezmi nasledujúcu $(n-k)$-ticu správy `message`: `message_part`
nájdi i také, že i-ty stĺpec H je rovný `message_part` − `syndrome`
if také i existuje **then**
`stream_part`$[i] \oplus = 1$
end if
end while
`stego_stream` + = `stream_part`
return `stego_stream`

nájdeme v skripte *MatrixEncoding.py*.

Algoritmus 7 Maticové extrahovanie

Vstup: kontrolná matica H,
vstupné dáta `stream`,
dĺžka kódu `len(part)`
Výstup: extrahovaná správa zo vstupu `message`
`message` =
while v dátach `stream` existujú bloky s neextrahovanou správou **do**
vezmi nasledujúci blok zo `stream` dĺžky `len(part)`, `part`
vypočítaj z neho syndróm `syndrome` = H.`part`^{T}
`message` + = `syndrome`
end while
return `message`

5.2.4 Zmena paketu

Maticové vkladanie schová tajnú správu medzi inými dátami. Algoritmus 8 popisuje vkladanie (ukrývanie) správy do UDP paketu, ktorý obsahuje Skype paket. Ukrývanie správy je založené na maticovom vkladaní do zašifrovaných dát, pričom zachovávame integritu Skype, UDP a IP paketu. Implementáciu nájdeme v skripte *ChangePacket.py*.

Algoritmus 8 Vkladanie správy do UDP paketu

Vstup: UDP paket `pkt`,
tajná správa `message`
Výstup: upravený paket so správou `changed_pkt`
z `pkt` vyber UDP dáta `data`
tretí byte dát `data` označ `function` = `data`[2]
prvé dva byty označ `frame_id` = `data`[0 : 2]
posledné dva byty dát označ `end` = `data`[−2 :]
zvyšok dát `data` tvoria zašifrované dáta `cipher_data` = `data`[3 : −2]
spočítaj $\texttt{crc}_1 = \texttt{CRC32}(\texttt{cipher\_data}||\texttt{end})$,
$\texttt{crc}_2 = \texttt{CRC32}(\texttt{cipher\_data})$,
$\texttt{tag} = \texttt{crc}_1 \oplus \texttt{frame\_id} \wedge \text{FFFF}_{16}$,
$\texttt{sequence\_num} = \texttt{crc}_2 \oplus \texttt{end} \wedge \text{FFFF}_{16}$
vlož správu do zašifrovaných dát:
$\texttt{new\_cipher\_data} = \texttt{MatrixEmb}(\texttt{cipher\_data}, \texttt{message})$
spočítaj $\texttt{new\_crc}_2 = \texttt{CRC32}(\texttt{new\_cipher\_data})$,
$\texttt{new\_end} = \texttt{sequence\_num} \oplus \texttt{new\_crc}_2 \wedge \text{FFFF}_{16}$,
$\texttt{new\_crc}_1 = \texttt{CRC32}(\texttt{new\_cipher\_data}||\texttt{new\_end})$,
$\texttt{new\_frame\_id} = \texttt{new\_crc}_1 \oplus \texttt{tag} \wedge \text{FFFF}_{16}$
$\texttt{new\_data} = \texttt{new\_frame\_id}||\texttt{function}||\texttt{new\_cipher\_data}||\texttt{new\_end}$
vytvor paket `changed_pkt` z pôvodnej hlavičky a z `new_data`
vypočítaj a prepíš kontrolné súčty v `changed_pkt` za aktuálne
return `changed_pkt`

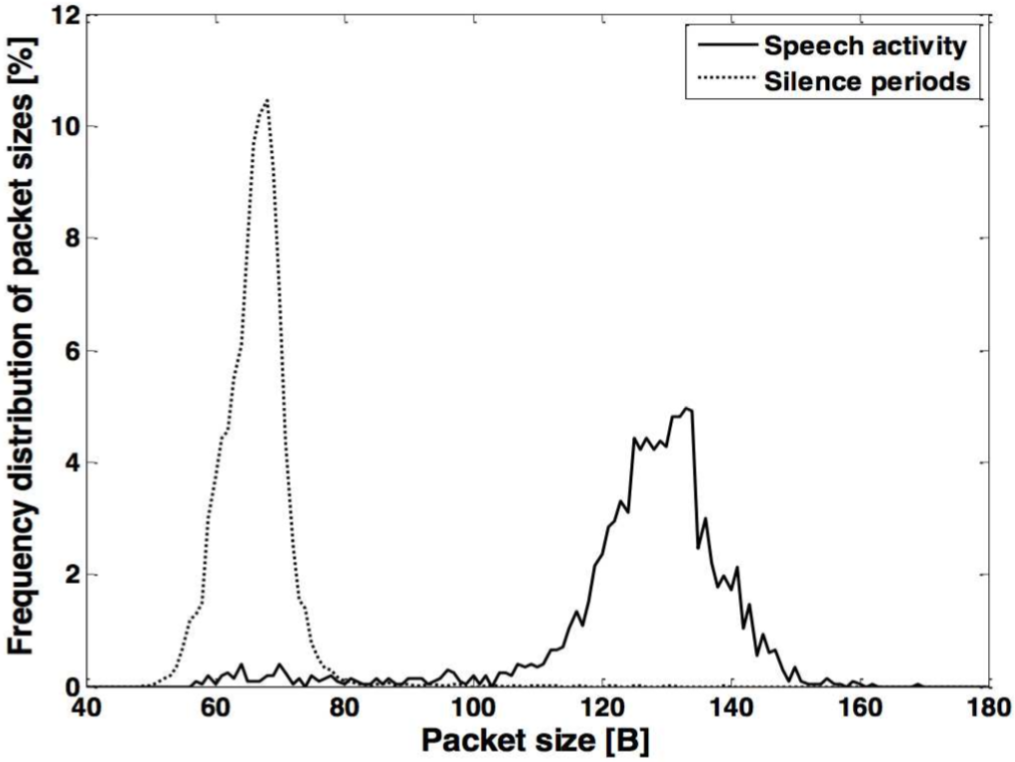
Pomocou pseudokódov sme popísali všetky algoritmy potrebné k samotnému vkladaniu (algoritmus 9) a extrahovaniu (algoritmus 10) správy do a z paketov počas hovoru. Začnime vkladaním.

5.2.5 Vkladanie správ do Skype hovoru

V aplikácii Skype zistíme, cez aký port komunikujeme. Vo verzii Skype 4.3.0.37 nájdeme port pod Menu → Options → Advanced. Odchytíme každý UDP paket, ktorý prejde týmto portom. Vyselektujeme z neho UDP dáta. Tie reprezentujeme binárne. Predpokladáme, že UDP dáta obsahujú Skype paket. Overíme, že ide o audio paket, teda funkcia je rovná 13_{10}. Ak áno, tak začneme vkladať správu do zašifrovaných dát.

Pozorovaním (odchytávaním skypových paketov) sme zistili, že najčastejšie sa počas rozhovoru vyskytujú pakety veľkosti od 90 do 150 bytov. K podobnému záveru prišli aj autori Mazurczyk, Karas a Szczypiorski [14]. Ako vidno na obrázku 5.3, pakety obsahujúce hlasové dáta sú veľkosti približne 100 až 160 bytov.

Každý blok vložíme do jedného paketu. Zvolili sme maticové vkladanie pomocou (7,4) = $(2^3 - 1, 4)$-kódu. Na každých 7 bitov nosiča vkladáme 3 bity správy, pričom zmeníme najviac jeden bit. Očakávame, že je nosič (dáta v pakete) veľkosti aspoň $7i$ a správa (vkladaná do jedného paketu) veľkosti aspoň $3i$, $i \in \mathbb{N}$. Ak je nosič veľkosti 100 bytov, tak pre veľkosť správy musí platiť, že je aspoň $(3 * 100)/7 = 42{,}86$ bytov. Samozrejme, nebudeme hľadať pakety veľkosti 100



Obr. 5.3: Veľkosť paketov.
Zdroj: [14].

bytov a iba do nich vkladať informácie. Veľkosť je len orientačná a to platí aj pre správu. Zvoľme, že do jedného paketu budeme vkladať 42 bytov informácie. Teda paket musí byť veľkosti aspoň $(7 * 42)/3 = 98$ bytov. Ak máme Hammingov $(2^j - 1, 2^j - 1 - j)$-kód a chceme vkladať 42 bytov informácie, tak musí platiť, že veľkosť paketu je aspoň $(42(2^j - 1))/j$ bytov, $j \in \mathbb{N}$. Aby sme vkladali efektívne, musíme j maximalizovať. Platí $(15 * 42)/4 = 157{,}5$. Pri vkladaní 42 bytov do paketu použitím $(7, 4)$-kódu sme najefektívnejší, ak bude paket veľkosti od 98 do 158 bytov. Pri 158 bytoch je výhodnejšie použiť $(15, 11)$-kód. Máme dve možnosti, budeme posielať správu len v paketoch veľkosti $98 - 158$ bytov, alebo v paketoch veľkosti aspoň 98 bytov. Zvolili sme druhú, menej efektívnu možnosť, aby sme urýchlili prenos.

Zašifrované dáta reprezentujeme binárne a tvoria náš nosič. Pri nezašifrovanom VoIP môže byť nosič tvorený sekvenciou najnižších bitov z každého bytu UDP dát. Tým zabezpečíme menšie poškodenie zvuku, viď vkladanie správ do audio signálu, sekcia 5.2.7.

Vkladanie pri našej implementácii prebieha tak, že správu rozdelíme do blokov po 39 bytov. Rozdelenie prebieha po „stĺpcoch“, ako sme už popísali vyššie. Prvé tri byty budú obsahovať úvodnú sekvenciu a poradové číslo. My sme zvolili pevnú úvodnú sekvenciu „IN␣“ zakódovanú pomocou T-kódov podľa tabuľky 3.1. Nevýhodu voľby pevnej sekvencie sme popísali vyššie. Vlastné generovanie úvodnej sekvencie sme neimplementovali. Zvyšných 13 bitov tvorí poradové číslo. Bloky správ vždy vkladáme do iných paketov a teda nevzniká žiadna podobnosť medzi paketmi. Úvodná sekvencia vyzerá v každom pakete (nosiči) inak. Môže sa stať, že úvodnú sekvenciu nájdeme aj v pakete, v ktorom sme neukryli žiadnu informá-

ciu. V takom prípade si to všimneme, poradové číslo bude podozrivé, napríklad príliš veľké. Počas testovania implementácie sme sa s náhodným vznikom úvodnej sekvencie nestretli.

Nevkladáme bloky správ do každého dostatočne veľkého paketu. Chceme, aby boli pakety, do ktorých budeme vkladať správu, od seba vzdialené aspoň o i, kde $i \in \mathbb{N}$. Preto vždy volíme $i : 20 \leq i \leq 50$.

Nech chceme poslať text, ten zakódujem pomocou T-kódov a následne pomocou Hammingovho kódu. Takto vzniknutý prúd bitov usporiadame do „stĺpcov“. Každý blok správy bude obsahovať 39 bytov.

Nasledujúci algoritmus 9 popisuje posielanie a skrývanie správ v UDP paketoch počas hovoru. Meníme routovaciu tabuľku tak, aby komunikácia na danom porte nebola posielaná adresátovi, ale bola poslaná do fronty na ďalšie spracovanie. Vo fronte pakety zmeníme a odtiaľ odošleme.

Algoritmus 9 Posielanie tajnej správy počas Skype hovoru

Vstup: port `port`,
tajná správa `secret_message`
zakóduj správu pomocou T-kódu a Hammingovho kódu:
`msg` = `HammEnc`(`TcodeEnc`(`secret_message`))
rozdeľ správu `msg` do blokov veľkosti 39 bytov
vytvor pravidlo pre routovaciu tabuľku, aby každý UDP paket `paket` na porte `port` vošiel do fronty na ďalšie spracovanie
poradové číslo paketu `num` = 1
while existuje paket vo fronte **and** existuje nespracovaný blok správy **do**
  if `funkcia` = 13_{10} **then**
    `counter` + = 1
    if `pkt` má aspoň 98 bytov **and** bloky správ sú vzdialené aspoň 20 paketov **then**
      `message` = `TcodeEnc`($IN\sqcup$)||`num`|| nasledujúci blok správy
      vlož správu `message` do `pkt`:
        `changed_pkt` = `ChangePacket`(`pkt`, `message`)
      pošli `changed_pkt`
      `num` + = 1
    end if
  end if
end while
vymaž vytvorené pravidlo z routovacej tabuľky a vypíš DONE.

5.2.6 Extrakcia správ zo Skype hovoru

Prijímanie a extrakcia správy z UDP paketov je jednoduchšia. Budeme odpočúvať pakety na porte, na ktorom pracuje naša aplikácia Skype. Keď do paketov vložíme všetky informácie, odosielateľ na obrazovke uvidí slovo „DONE.“. To znamená, že je čas ukončiť rozhovor (alebo aspoň odchytávanie paketov). Príjemca má všetky pakety (až na tie stratené) potrebné k dekódovaniu správy.

Prezrieme každý odchytený UDP paket. Ak je funkcia 13_{10}, budeme hľadať úvodnú sekvenciu „IN$\sqcup$“. Ak ju nájdeme, tak pomocou maticovej extrakcie vyse-

lektujeme poradové číslo paketu a blok správy. Po prejdení všetkých odchytených paketov zoradíme bloky správy podľa čísla paketu. Chýbajúce pakety nahradíme blokom 39 bytov núl. Jednotlivé bloky zreťazíme a zotriedime zo „stĺpcov“. Následne dekódujeme pomocou Hammingových kódov a T-kódov. Získali sme poslanú správu. Posielanie a prijímanie paketov nájdeme v *SendingPackets.py* a *ReceivingPackets.py*. Skripty ukončíme pomocou keyboard interrupt (Ctrl + c).

Algoritmus 10 Prijímanie tajnej správy počas Skype hovoru

Vstup: port `port`
Výstup: tajná správa `secret_message`

```
pairs = {}
while nemáme všetky pakety do
  odpočúvaj UDP pakety na porte port
  if funkcia = 13_10 and veľkosť paketu je aspoň 98 bytov then
    zo zašifrovaných dát UDP paketu vyber správu msg pomocou maticovej
    extrakcie
    if TcodeEnc(prvých 11 bitov msg) = IN␣ then
      nasledujúcich 13 bitov je num a zvyšok msg_block
      pairs+ =(num, msg_block)
    end if
  end if
end while
zotrieď pairs podľa num
msg_blocks = zreťazené msg_block z pairs
message = transformácia msg_blocks zo stĺpcov do pôvodnej podoby
secret_message = TcodeDec(HammDec(message))
```

Algoritmy nájdeme v priloženom CD (obsah v prílohe 5.2.7). Sú naprogramované v jazyku Python. Pred spustením je potrebné mať nainštalované Scapy a Python spolu s balíčkom nfqueue. Scapy slúži na prácu s paketami, nfqueue na prácu s routovacími tabuľkami. Aby sme mohli meniť routovacie tabuľky, potrebujeme mať administrátorské práva. Používali sme Python verzie 2.7, Scapy verzie 2.2 a nfqueue-bindings 0.3-4build1. Testované na systéme Ubuntu 12.04.

5.2.7 Vkladanie správ do audio signálu

Navrhnutú steganografickú metódu môžeme adaptovať aj na vkladanie tajných správ do digitálneho audio signálu. Pre jednoduchosť predpokladajme, že signál neprechádza kompresiou.

Nech $(p_1, \ldots, p_n) \in \mathcal{G}^n$ je reprezentovaný audio signál v digitálnej podobe bez straty informácie, kde $n \in \mathbb{N}$ a $\mathcal{G}$ je množina všetkých možných hodnôt danej digitálnej reprezentácie. Tak, ako sme zadefinovali v podkapitole 2.4, majme funkciu s a pomocou nej vytvorme binárny nosič

$$\mathbf{x} = (x_1, \ldots, x_n) = (\mathrm{s}(p_1), \ldots, \mathrm{s}(p_n)) \in \mathbb{F}_2^n.$$

Nosič môže byť tvorený napríklad najmenej významnými bitmi.

Obdobne, ako pri vkladaní správy do paketov, chceme vložiť do nosiča tajnú správu. Tú najprv skomprimujme pomocou T-kódov a potom ju maticovo vložme do nosiča.

Nech vloženie správy do nosiča vyžaduje zmenu bitu $x_i, 1 \leq i \leq n$. To pre nás bude znamenať zmenu hodnoty p_i napríklad pomocou LSB matchingu. Tým zaručíme nepostrehnuteľnú zmenu v audio signále.

Záver

V tejto práci sme navrhli a naimplementovali steganografickú metódu, ktorá ukrýva informácie v zašifrovaných dátach Skype paketu. Pri navrhovaní steganografického systému sme sa zamerali na odolnosť voči pasívnemu útočníkovi. Zároveň sme do modelu zahrnuli aj poznatky z reálnych sietí a naimplementovali model odolný voči nespoľahlivému prenosovému spojeniu. Tajnú správu sme rozdelili do rovnako veľkých blokov po „stĺpcoch“ a každý blok správy prenášali v jednom pakete. Na prerozdelenie bitov do stĺpcov sme pozerali ako na permutáciu, ktorá je známa len zainteresovaným stranám. Ďalej sme zanalyzovali, ako vhodne voliť permutáciu, aby sme zachovali odolnosť steganografického systému voči nespoľahlivému prenosu.

Aby sme dokázali zrekonštruovať pôvodnú tajnú správu aj pri strate paketov, tak správu kódujeme pomocou T-kódov a následne Hammingovými kódmi. T-kódy sú odvodené od Huffmanových kódov a komprimujú tajnú správu.

Na rozpoznávanie paketov obsahujúcich tajnú informáciu a na ich správne poradie, spolu s blokom správ maticovo vkladáme aj úvodnú sekvenciu a poradové číslo paketu. Implementácia je s pevnou úvodnou sekvenciou. Poradové číslo slúži aj k tomu, aby sme spoznali, ktoré pakety sa stratili. Na zvýšenie bezpečnosti sme navrhli vylepšenie s generovaním úvodnej sekvencie závislej na predom dohodnutom kľúči a poradovom čísle Skype paketu.

Navrhnutú steganografickú metódu sme úspešne naimplementovali a otestovali v reálnych podmienkach. Zoznámili sme sa pri tom s internetovými protokolmi IP a UDP a s prácou s routovacími tabuľkami.

Popísali sme adaptovanie navrhnutého steganografického systému na vkladanie tajnej správy do audio signálu. Predpokladali sme pri tom „ideálny“ audio signál, teda taký, ktorý nie je ovplyvňovaný žiadnym stratovým kodekom a ani inou manipuláciou signálu.

Možné budúce rozšírenie vidíme v doimplementovaní variabilnej úvodnej sekvencie a v zlepšení užívateľského prostredia. Prípadne je do praktického použitia vhodné tajnú správu pred vkladaním zašifrovať.

Zoznam použitej literatúry

[1] Aoki, N. (2008). A technique of lossless steganography for G.711 telephony speech. In *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 608–611. IEEE.

[2] BOHÁČEK, M. (2012). Analýza voice over IP protokolů. Master's thesis, Matematicko-fyzikální fakulta, Univerzita Karlova, Praha.

[3] Daemen, J. a Rijmen, V. (2013). *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media.

[4] Davidson, J., Peters, J. a Gracely, B. (2000). *Voice Over IP Fundamentals*. Cisco Press fundamentals series. Cisco Press, Indianapolis. ISBN 978-1-578-70168-1.

[5] Djebbar, F., Ayad, B., Hamam, H. a Abed-Meraim, K. (2011). A view on latest audio steganography techniques. In *Innovations in Information Technology (IIT), 2011 International Conference on*, pages 409–414. IEEE.

[6] Fridrich, J. (2009). *Steganography in digital media: principles, algorithms, and applications*. Cambridge University Press.

[7] Fridrich, J., LISONĚK, P. a Soukal, D. (2007). On steganographic embedding efficiency. In *Information Hiding*, pages 282–296. Springer.

[8] Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell System Technical Journal*, **29**(2), 147–160.

[9] Lubacz, J., Mazurczyk, W. a Szczypiorski, K. (2010). Voice over IP. *Spectrum, IEEE*, **47**(2), 42–47.

[10] Manoharan, S. (2003). Towards robust steganography using T-codes. In *Video/Image Processing and Multimedia Communications, 2003. 4th EURASIP Conference focused on*, volume 2, pages 707–711. IEEE.

[11] Manoharan, S. a Mitra, S. (2009). Message recovery enhancements to LSB embedding and echo hiding based on T-codes. In *Communications, Computers and Signal Processing, 2009. PacRim 2009. IEEE Pacific Rim Conference on*, pages 215–220. IEEE.

[12] Mazurczyk, W. a Lubacz, J. (2010). LACK—a VoIP steganographic method. *Telecommunication Systems*, **45**(2-3), 153–163.

[13] Mazurczyk, W. a Szczypiorski, K. (2008). Steganography of VoIP streams. In *On the Move to Meaningful Internet Systems: OTM 2008*, pages 1001–1018. Springer.

[14] Mazurczyk, W., Karas, M. a Szczypiorski, K. (2013). SkyDe: A Skype-based steganographic method. *arXiv preprint arXiv: 1301.3632*.

[15] MISTRÍK, J. (1966). Dĺžka slova a štylistická štruktúra textu. *Jazykovedný časopis*, **17**, 113–120.

[16] Nicolescu, R. a Titchener, M. R. (1998). Uniqueness theorems for T-codes. *Romanian Journal of Information Science and Technology*, **1**(3), 243–258.

[17] Ould Medeni, M. B. a Souidi, E. M. (2010). A steganography schema and error-correcting codes. *Journal of Theoretical and Applied Information Technology*, **18**(1), 42–47.

[18] Purser, M. (1995). *Introduction to Error-correcting Codes*. Artech House telecom library. Artech House, Norwood. ISBN 978-0-890-06784-0.

[19] Rosen, K. H. (2012). *Discrete Mathematics and Its Applications*, chapter Coding Theory. McGraw-Hill Higher Education, 7 edition.

[20] ŠTEFÁNIK, J., Rusko, M. a POVAŽANEC, D. (1999). Frekvencia slov, grafém, hlások a ďalších elementov slovenského jazyka. *Jazykovedný časopis*, **50**(2), 81–93.

[21] Titchener, M. R. (1985). Construction and properties of the augmented and binary-depletion codes. *IEE Proceedings*, **132**(3), 163–169.

[22] Titchener, M. R. (1996). Generalised T-codes: extended construction algorithm for self-synchronising codes. *IEE Proceedings-Communications*, **143**(3), 122–128.

[23] Titchener, M. R. (2008). An introduction to T-codes. `http://tcode.auckland.ac.nz/~mark/T-codes%3A%20Intro.html`. Posledný prístup: 10.05. 2017.

[24] Van Lint, J. H. (2012). *Introduction to Coding Theory*, volume 86 of *California Institute of Technology*. Springer Science & Business Media, Pasadena. ISBN 978-3-642-63653-0.

Zoznam obrázkov

Zoznam algoritmov

Prílohy

Obsah priloženého CD:

- TcodeEncoding.py
- HammingEncoding.py
- MatrixEncoding.py
- ChangePacket.py
- SendingPackets.py
- ReceivingPackets.py